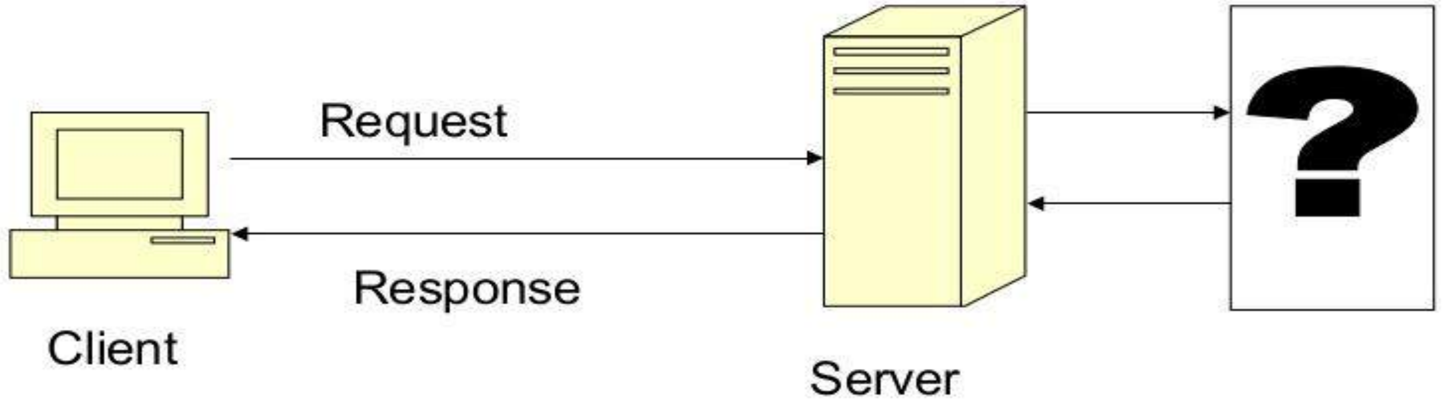
Servlets

TOPICS TO BE DISCUSSED

- WEB APPLICATION
- WAR FILE
- SERVERS
- PORTS
- SERVERSIDE PROGRAMMING
- WEB SERVER/APPLICATION SERVER
- CGI SCRIPT
- SERVLET
- CGI vs SERVLET
- SERVLET CONTAINER
- SERVLET PACKAGE
- SERVLET LIFE CYCLE
- FIRST SERVLET

Web Application

- What is web application?
  - Application that runs on WWW is known as web application.
- How does it work?



War File

- A **war (web archive) File** contains files of a web project. It may have servlet, xml, jsp, image, html, css, js etc. files.
- What is war file?
- web archive (war) file contains all the contents of a web application. It reduces the time duration for transferring file.

Servers

- A server is a computer that responds to requests from a client
  - Typical requests: provide a web page, upload or download a file, send email
- A server is also the software that responds to these requests; a client could be the browser or other software making these requests
- Typically, your little computer is the client, and someone else's big computer is the server
  - However, any computer can be a server
  - It is not unusual to have server software and client software running on the same computer

Apache

- Apache is a *very* popular server
  - 66% of the web sites on the Internet use Apache
- Apache is:
  - Full-featured and extensible
  - Efficient
  - Robust
  - Secure (at least, more secure than other servers)
  - Up to date with current standards
  - Open source
  - Free

Tomcat

- Tomcat is the Servlet Engine than handles servlet requests for Apache
  - Tomcat is a "helper application" for Apache
  - It's best to think of Tomcat as a "servlet container"
- Apache can handle many types of web services
  - Apache can be installed without Tomcat
  - Tomcat can be installed without Apache
- It's easier to install Tomcat standalone than as part of Apache
  - By itself, Tomcat can handle web pages, servlets, and JSP
- Apache and Tomcat are open source (and therefore free)

Ports

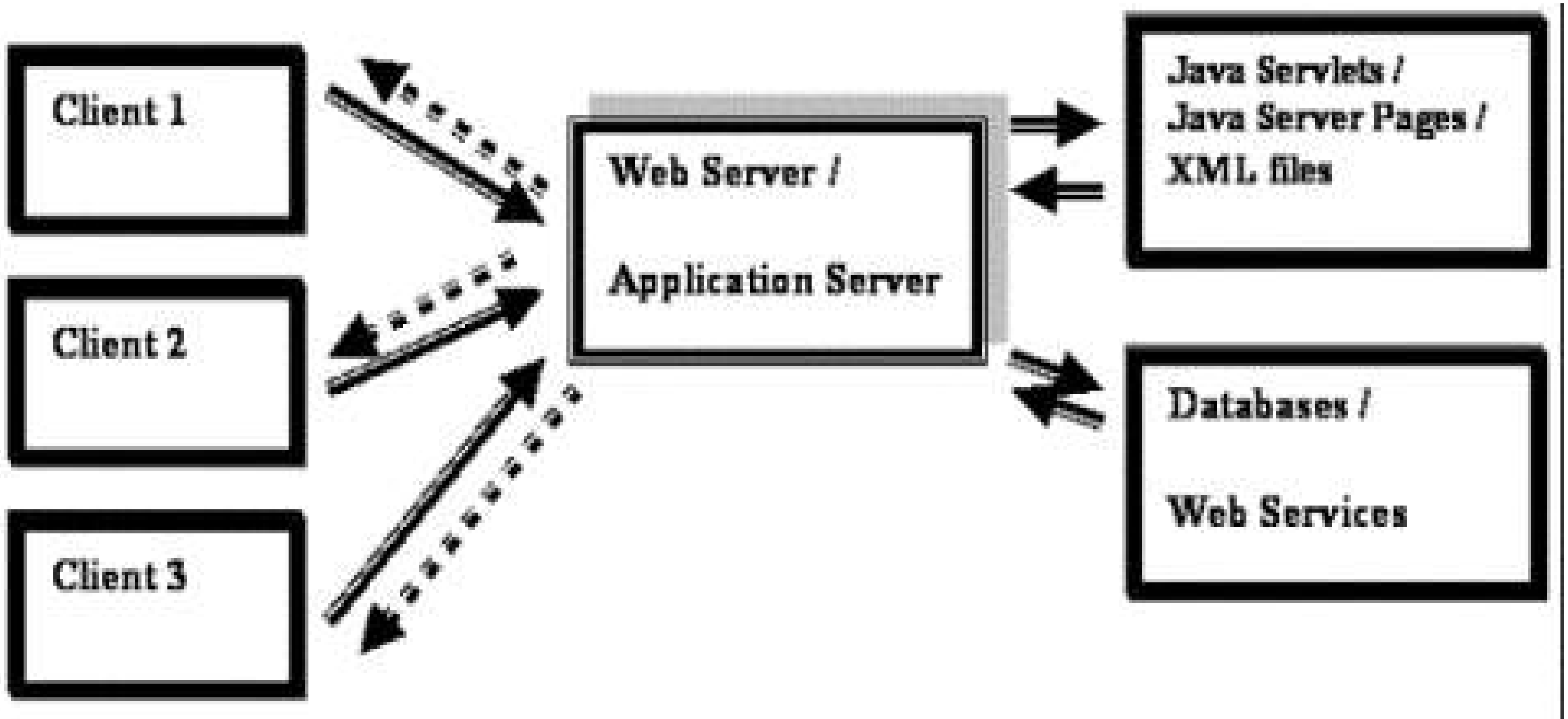
- A port is a connection between a server and a client
  - Ports are identified by positive integers
  - A port is a software notion, not a hardware notion, so there may be very many of them
- A service is associated with a specific port
  - Typical port numbers:
    - 21—FTP, File Transfer Protocol
    - 3306 —SQL SERVER
    - 25—SMTP, Simple Mail Transfer Protocol
    - 53—DNS, Domain Name Service
    - 80—HTTP, Hypertext Transfer Protocol
    - 8080—HTTP (used for testing HTTP)

$\}$ These are the ports of most interest to us

Why Server Side Programming?

- Though it is technically feasible to implement almost any business logic using client side programs, logically or functionally it carries no ground when it comes to enterprise applications (e.g. banking, air ticketing, e-shopping etc.).
- To further explain, going by the client side programming logic; a bank having 10,000 customers would mean that each customer should have a copy of the program(s) in his or her PC which translates to 10,000 programs!
- In addition, there are issues like security, resource pooling, concurrent access and manipulations to the database which simply cannot be handled by client side programs. The answer to most of the issues cited above is ?
- Server Side Programming?.

Server Side Programming



Fieure – Server Side Proerammine (architecture)

Advantages of Server Side Programs

I. All programs reside in one machine called the Server. Any number of remote machines (called clients) can access the server programs.

II. New functionalities to existing programs can be added at the server side which the clients? can advantage without having to change anything from their side.

III. Migrating to newer versions, architectures, design patterns, adding patches, switching to new databases can be done at the server side without having to bother about clients hardware or software capabilities.

Advantages of Server Side Programs Cont..

I. Issues relating to enterprise applications like resource management, concurrency, session management, security and performance are managed by service side applications.

II. They are portable and possess the capability to generate dynamic and user-based content (e.g. displaying transaction information of credit card or debit card depending on user?s choice).

Advantages of Server Side Programs Cont ...

- The **server-side extensions** are nothing but the technologies that are used to create dynamic Web pages. Actually, to provide the facility of dynamic Web pages, Web pages need a container or Web server. To meet this requirement, independent Web server providers offer some proprietary solutions in the form of **APIs**(Application Programming Interface).
- These **APIs** allow us to build programs that can run with a Web server. In this case , **Java Servlet** is also one of the component APIs of **Java Platform Enterprise Edition** which sets standards for creating dynamic Web applications in Java

Server: Web vs. Application

- Server is a device or a computer program that accepts and responds to the request made by other program, known as client. It is used to manage the network resources and for running the program or software that provides services.
- **There are two types of servers:**
- Web Server
- Application Server

Web Server

- Web server contains only web or servlet container. It can be used for servlet, jsp, struts, jsf etc. It can't be used for EJB.
- It is a computer where the web content can be stored. In general web server can be used to host the web sites but there also used some other web servers also such as FTP, email, storage, gaming etc.
- Examples of Web Servers are:
- **Apache Tomcat** and **Resin**.

Web Server

- A web server is the combination of computer and the program installed on it. Web server interacts with the client through a web browser. It delivers the web pages to the client and to an application by using the web browser and the HTTP protocols respectively.
- We can also define the web server as the package of large number of programs installed on a computer connected to Internet or intranet for downloading the requested files using File Transfer Protocol, serving e-mail and building and publishing web pages.

Application Server

- Application server contains Web and EJB containers. It can be used for servlet, jsp, struts, jsf, ejb etc. It is a component based product that lies in the middle-tier of a server centric architecture.
- It provides the middleware services for state maintenance and security, along with persistence and data access. It is a type of server designed to install, operate and host associated services and applications for the IT services, end users and organizations.

The Example of Application Servers are:

- **JBoss**: Open-source server from JBoss community.
- **Glassfish**: Provided by Oracle.
- **Weblogic**: Provided by Oracle. It more secured.
- **Websphere**: Provided by IBM.

CGI Scripts

- CGI stands for “Common Gateway Interface”

Client sends a request to server

Server starts a CGI script

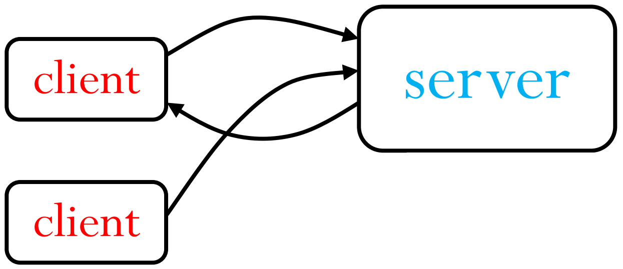
Script computes a result for server and quits

Server returns response to client

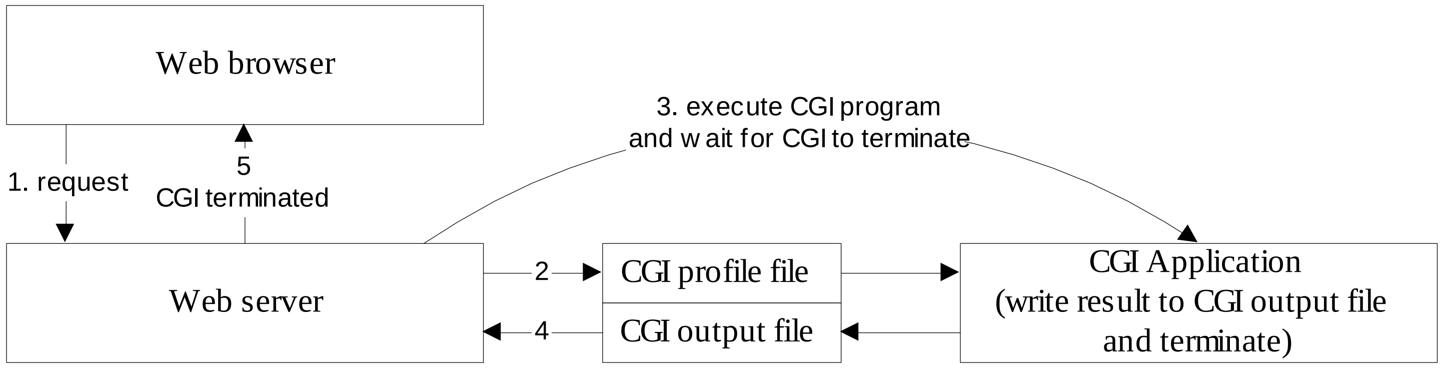
Another client sends a request

Server starts the CGI script again

Etc.



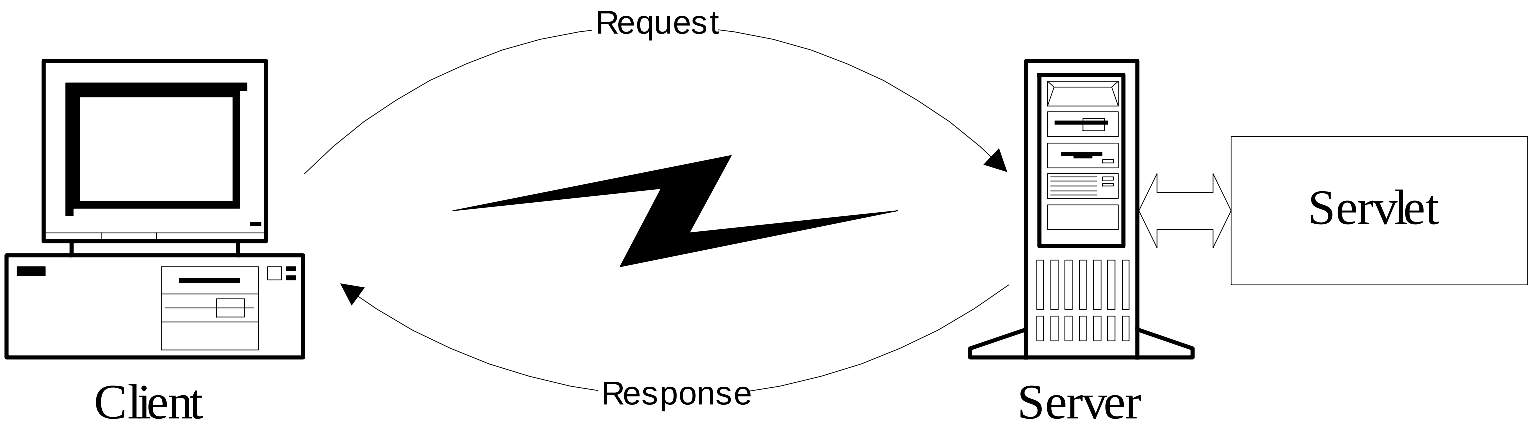
CGI Communication (traditional approach)


Web browser
1. request
5
CGI terminated
3. execute CGI program and w ait for CGI to terminate
Web server
2
4
CGI profile file
CGI output file
CGI Application (write result to CGI output file and terminate)

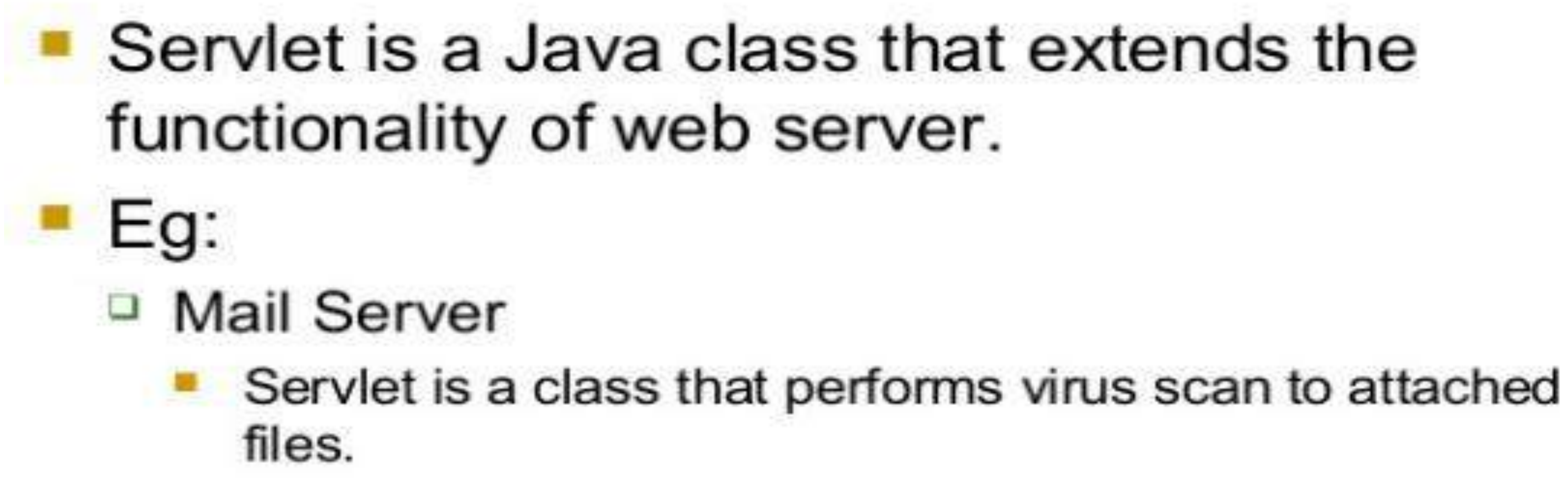
1. Web browser requset a response from CGI program.
2. Web server create CGI profile file that was contain information about CGI variable, server and CGI output file.
3. Web server starts CGI application and wait for its termination.
4. CGI program runs and writes the result to CGI output file and then terminates.
5. Web server reads from CGI output file and send back to Web browser.

What is java servlet ?

A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.



What is Servlet cont ...



- Servlet is a Java class that extends the functionality of web server.
- Eg:
  - Mail Server
    - Servlet is a class that performs virus scan to attached files.

Example use of Servlet

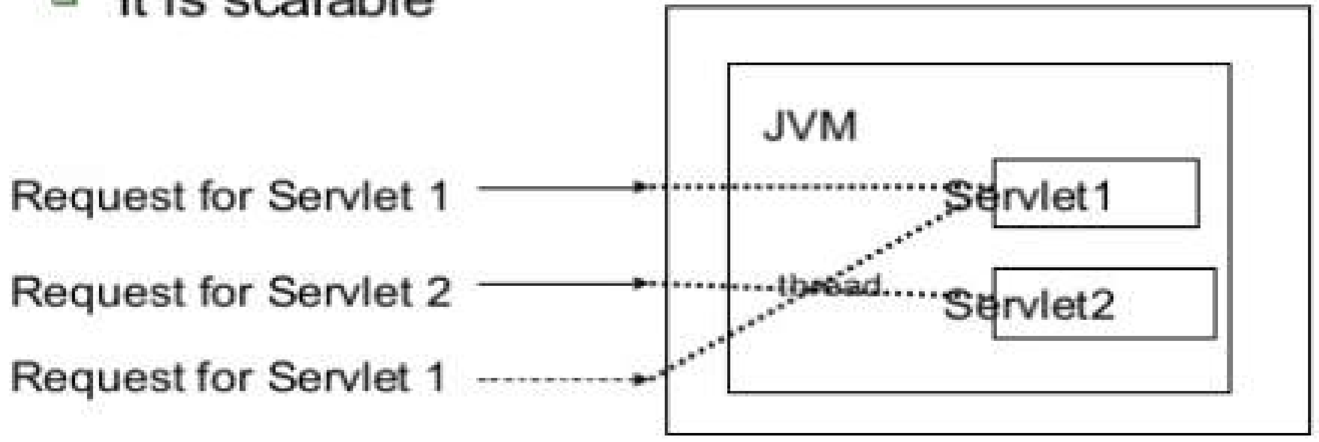
- Processing data POST over HTTPs using HTML form as purchase order or credit card data
- Allowing collaborative between people such as on-line conferencing
- Many servlet can chain together among Web servers to reduce load on one Web server.

Why Use Servlets ?

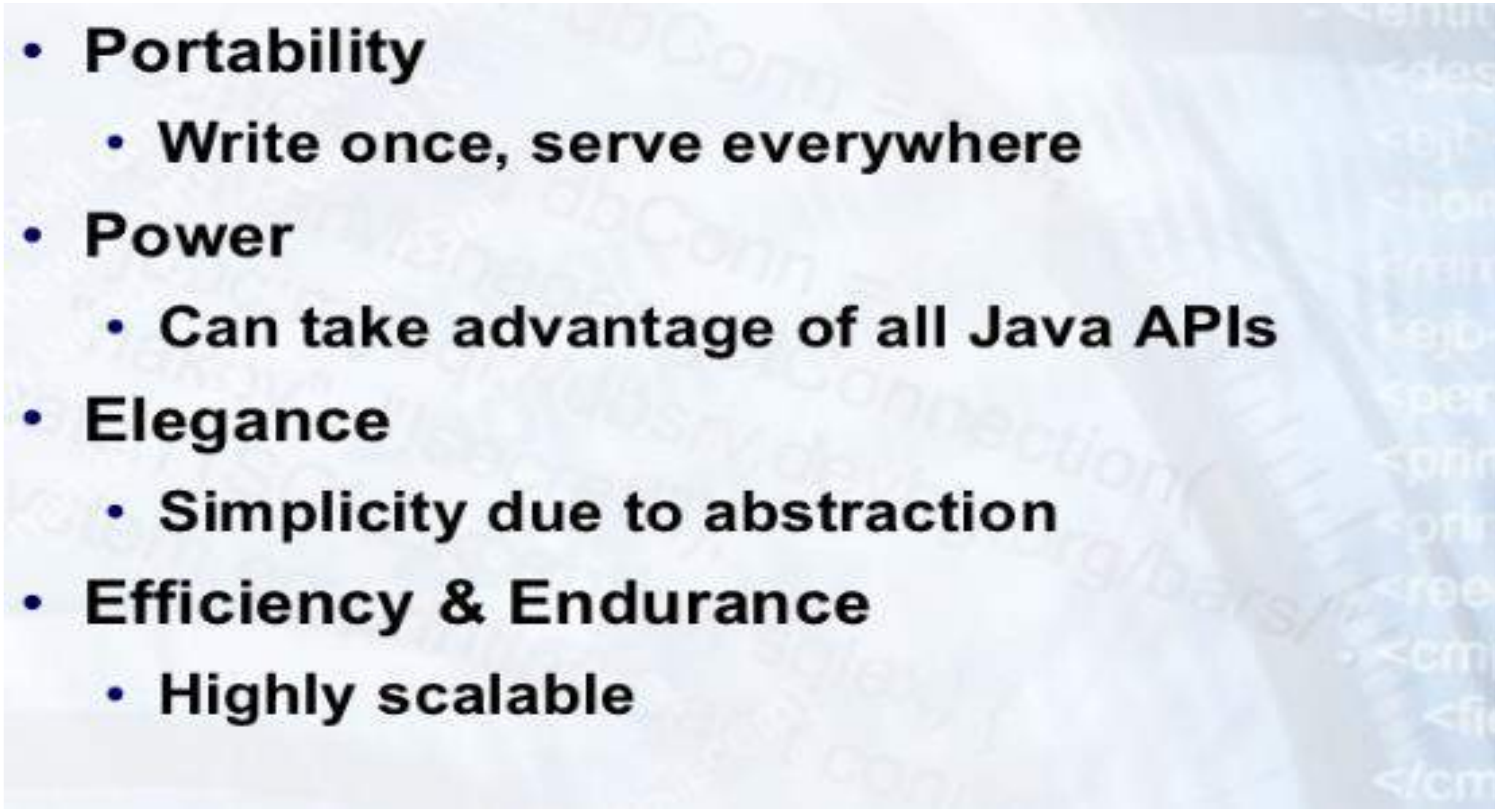
- One of the reasons that Java became popular is Applets.
  - but there are problems with Applets
    - Browser compatibility
    - Download bandwidth
- Server-side Java
  - the code is executed on the server side not the client side
  - a dynamically loaded module that services requests from a Web server

Benefits

- Only one copy of servlet is loaded into JVM.
- Each request begins a new thread to servlet than a process.
  - Saves time
  - Response time increases
  - It is scalable

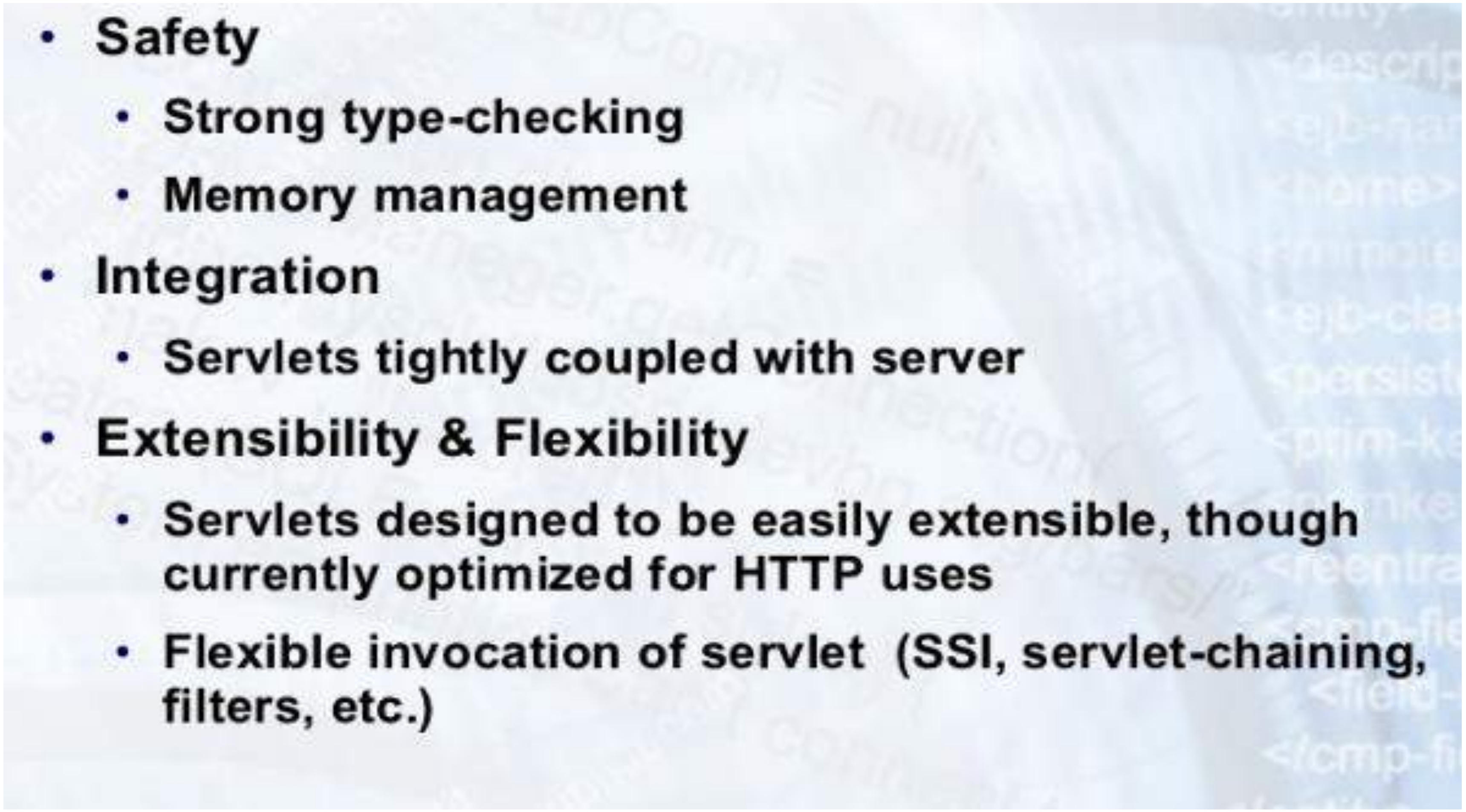


Why Use Servlet



- **Portability**
  - **Write once, serve everywhere**
- **Power**
  - **Can take advantage of all Java APIs**
- **Elegance**
  - **Simplicity due to abstraction**
- **Efficiency & Endurance**
  - **Highly scalable**

Why use Servlet Cont …



- **Safety**
  - **Strong type-checking**
  - **Memory management**
- **Integration**
  - **Servlets tightly coupled with server**
- **Extensibility & Flexibility**
  - **Servlets designed to be easily extensible, though currently optimized for HTTP uses**
  - **Flexible invocation of servlet (SSI, servlet-chaining, filters, etc.)**

Servlets

- A servlet is like an applet, but on the server side

Client sends a request to server

Server starts a servlet

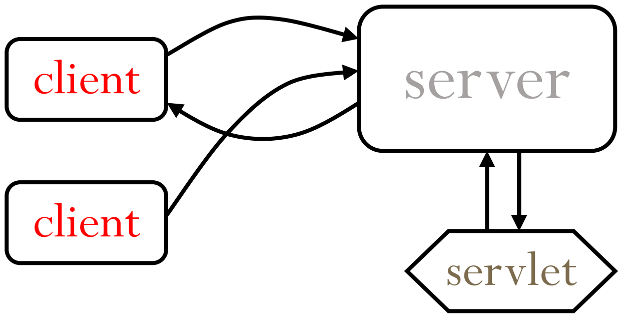
Servlet computes a result for server and *does not quit*

Server returns response to client

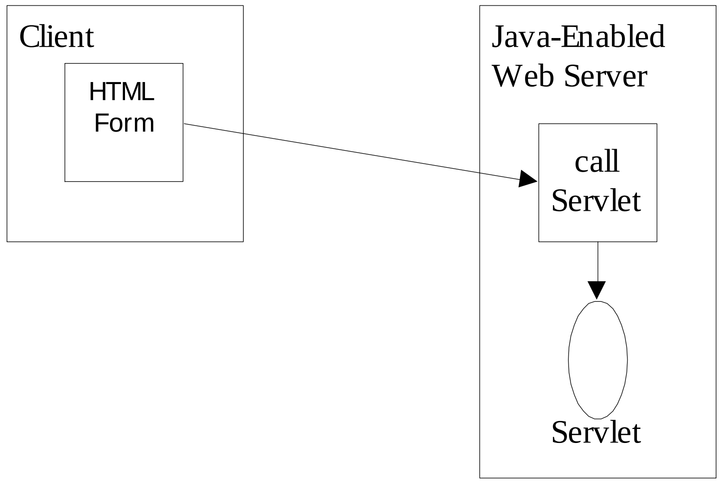
Another client sends a request

Server calls the servlet again

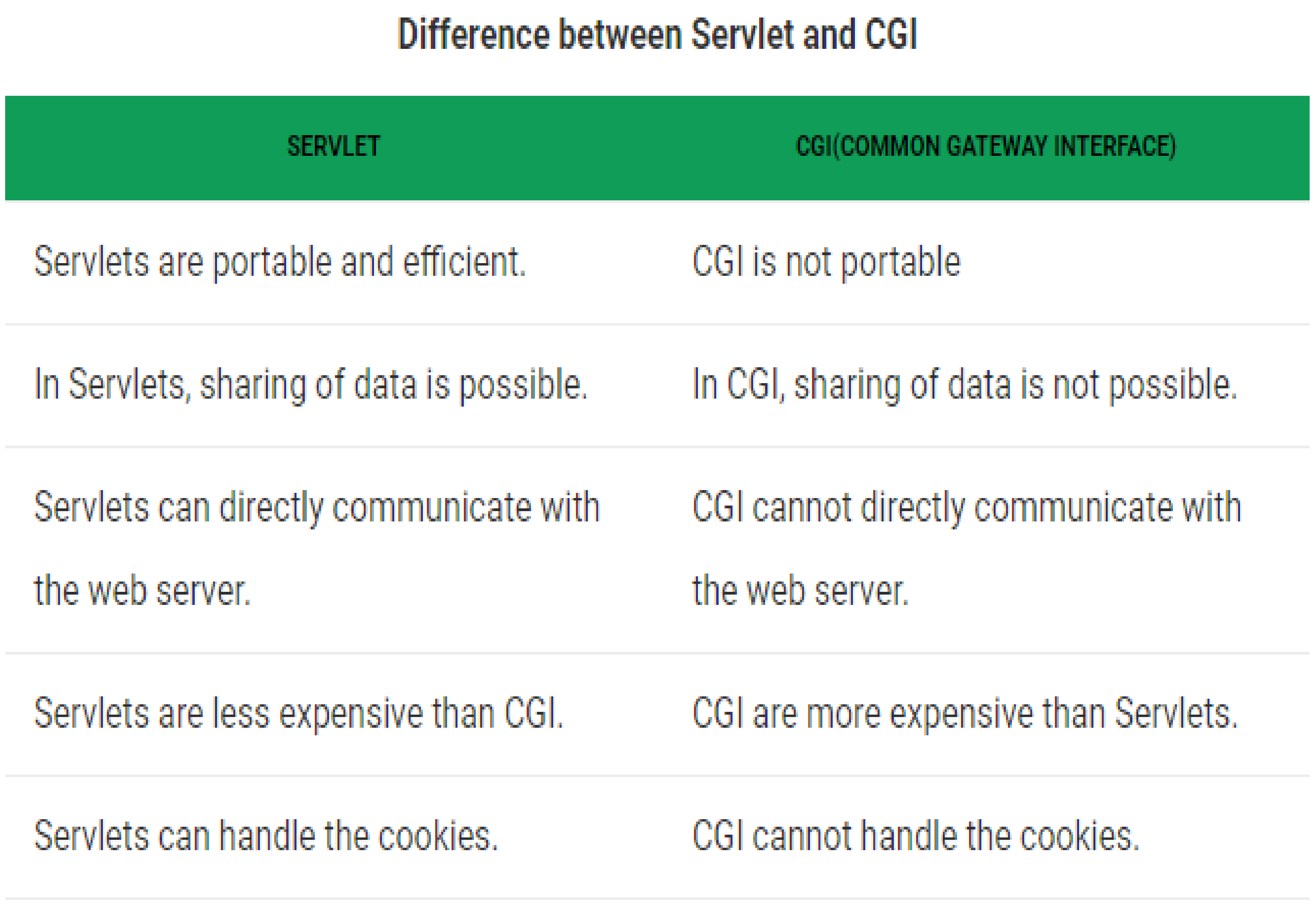
Etc.



Servlet Communication



- Web browser request servlet by specified URL as http://www.host.com/serlet/servletName
- Web server call service() method in ServletLoader class which will dynamically load a specified servlet name from a special directory call *servlet.*



Difference between Servlet and CGI

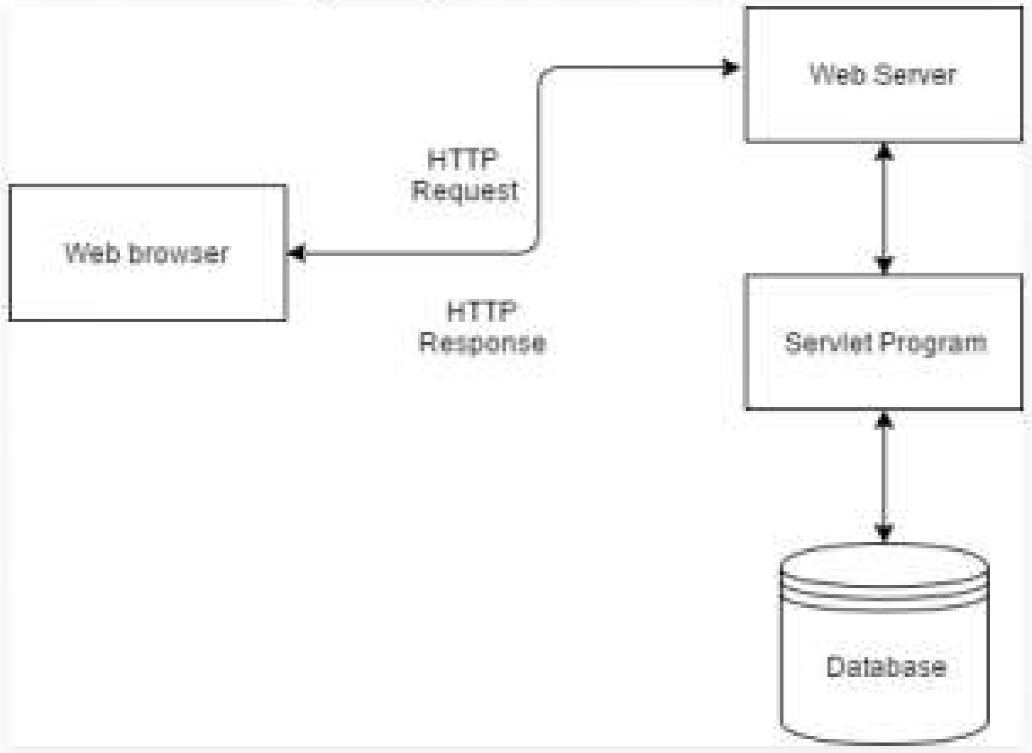
SERVLET	CGI(COMMON GATEWAY INTERFACE)
Servlets are portable and efficient.	CGI is not portable
In Servlets, sharing of data is possible.	In CGI, sharing of data is not possible.
Servlets can directly communicate with the web server.	CGI cannot directly communicate with the web server.
Servlets are less expensive than CGI.	CGI are more expensive than Servlets.
Servlets can handle the cookies.	CGI cannot handle the cookies.

Servlets (contd.)

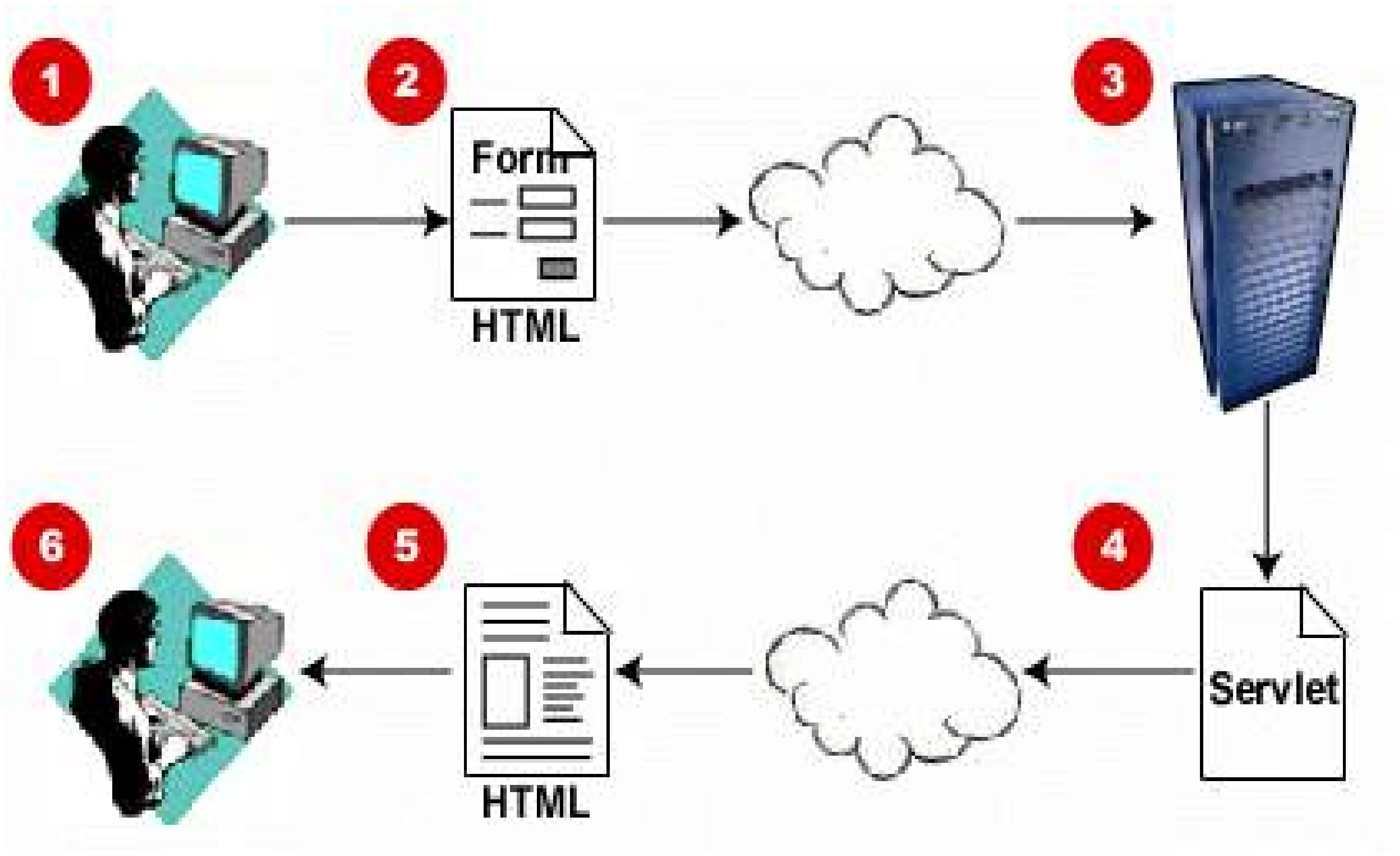
- vs. Server-Side JavaScript
  - only available on certain web servers
- vs. Active Server Pages (ASP)
  - only available on certain web servers

Servlet Architecture

The following diagram shows the servlet architecture:



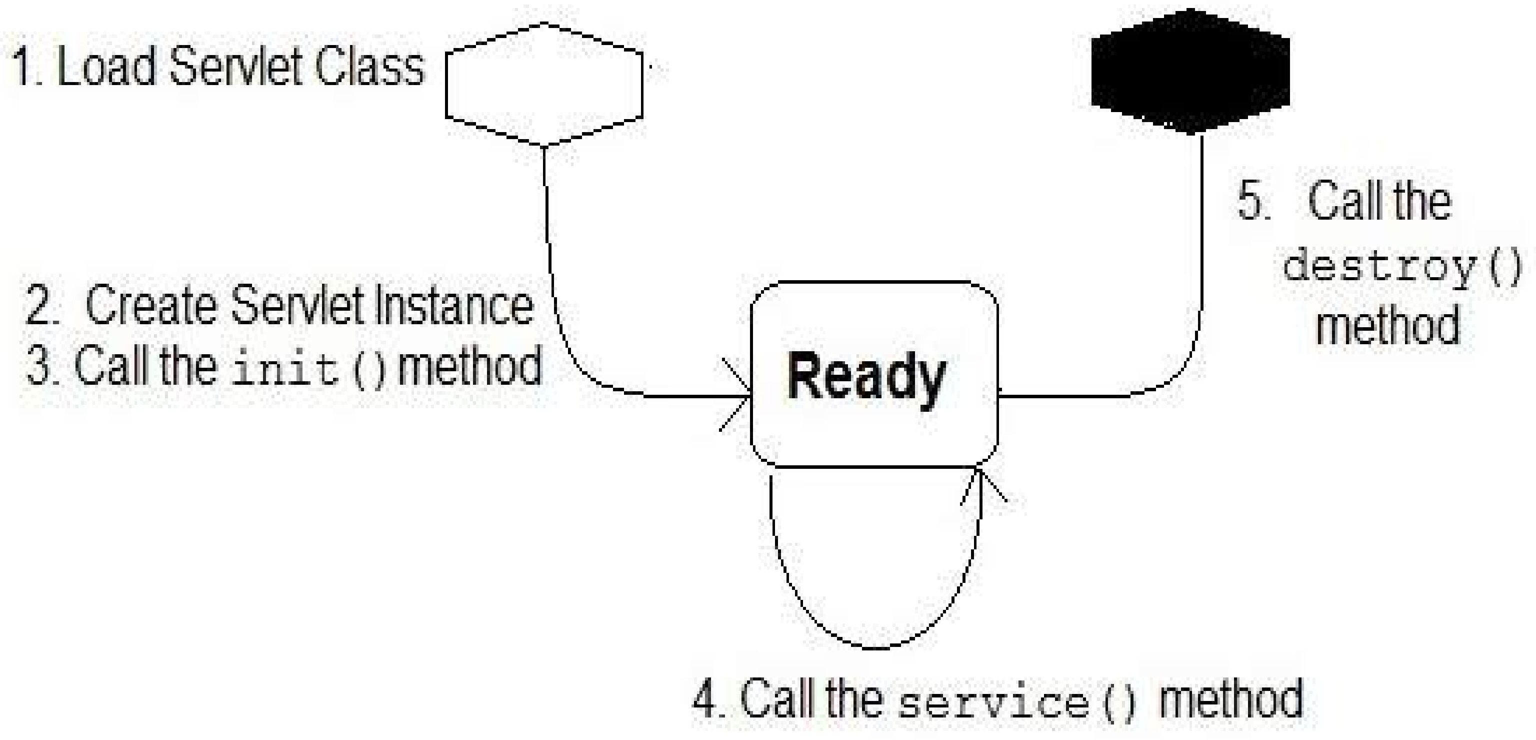
How Servlet Works



Life Cycle of a Servlet

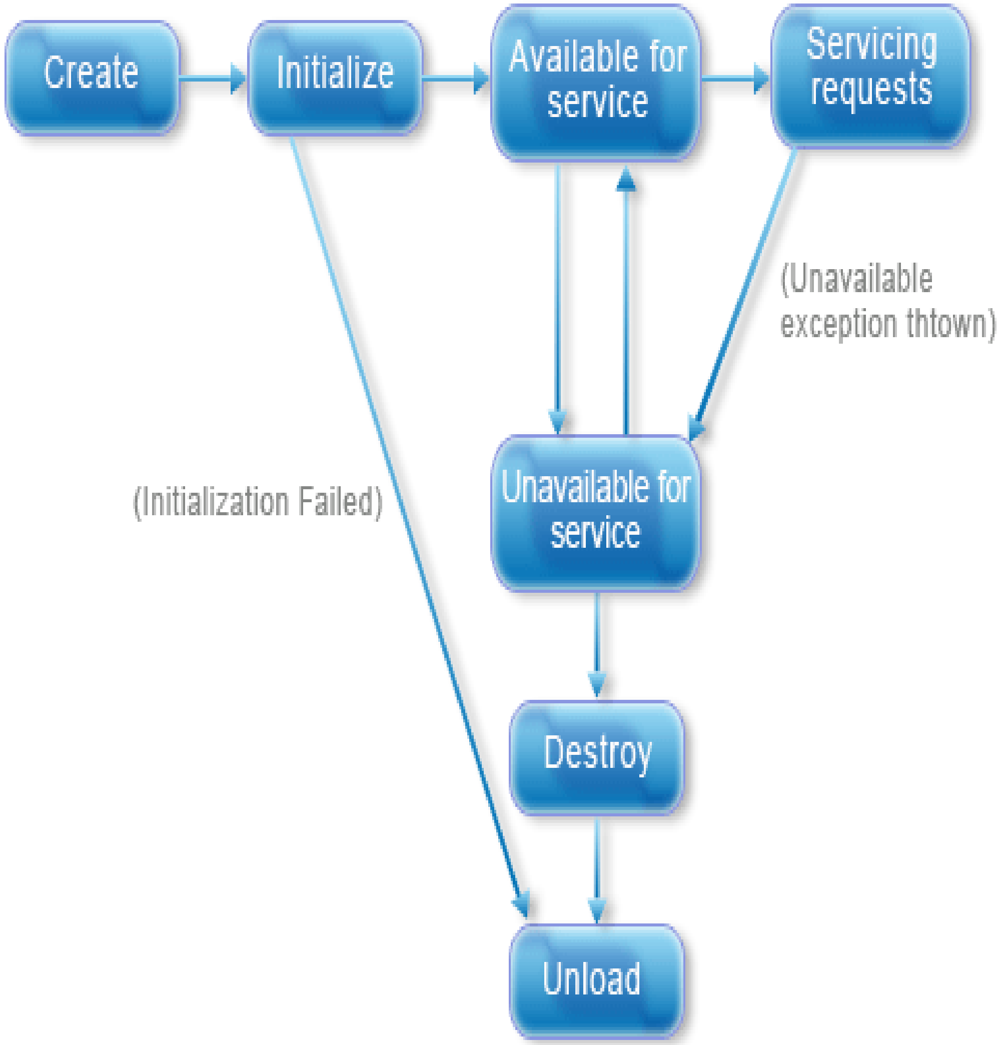
- The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:
  1. Servlet class is loaded.
  2. Servlet instance is created.
  3. init method is invoked.
  4. service method is invoked.
  5. destroy method is invoked.

Life cycle of Servlet

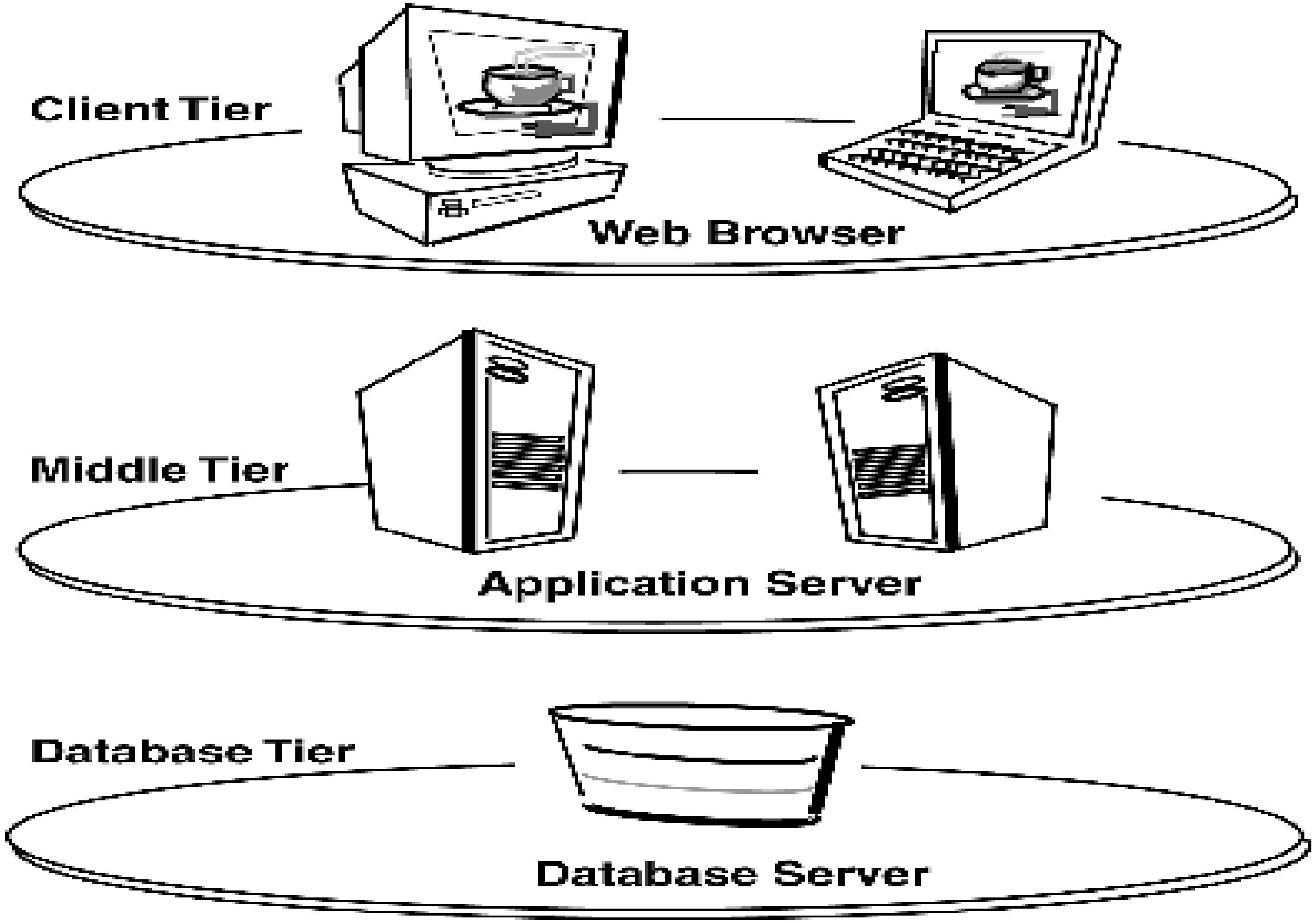

1. Load Servlet Class
2. Create Servlet Instance
3. Call the init() method
Ready
4. Call the service() method
5. Call the destroy() method

Servlet Life Cycle

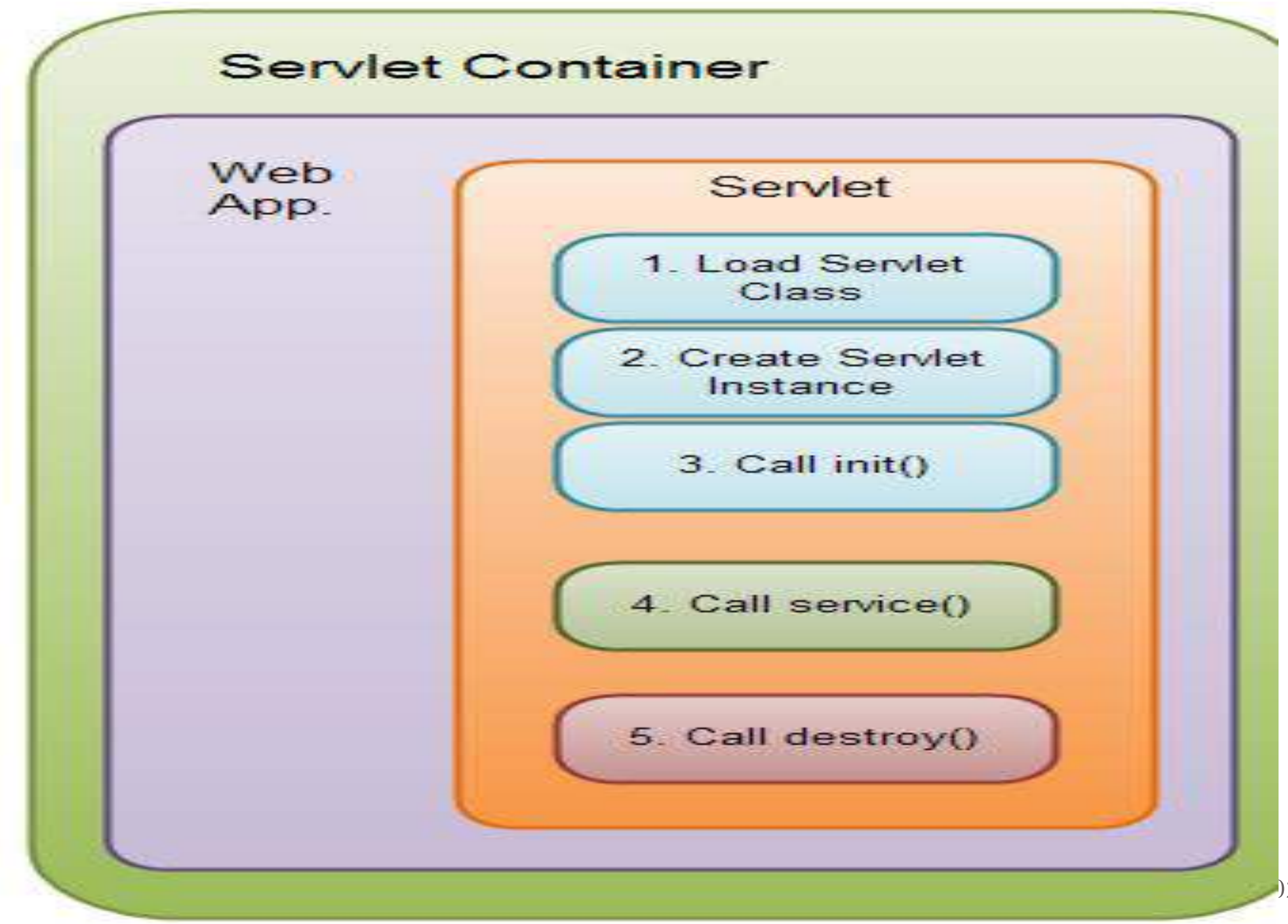
- The life cycle of the servlet is controlled by the servlet container. The servlet container is responsible for doing following task
- When the first request is made for servlet the containet loads the servlet class and initiates it.
- Then make an instance of servlet class.
- Initializes the servlet class by calling init() method.
- Call the service method passes the request response object to the service method.
- If container need to remove the servlet then call the destroy() method to finalize the servlet.

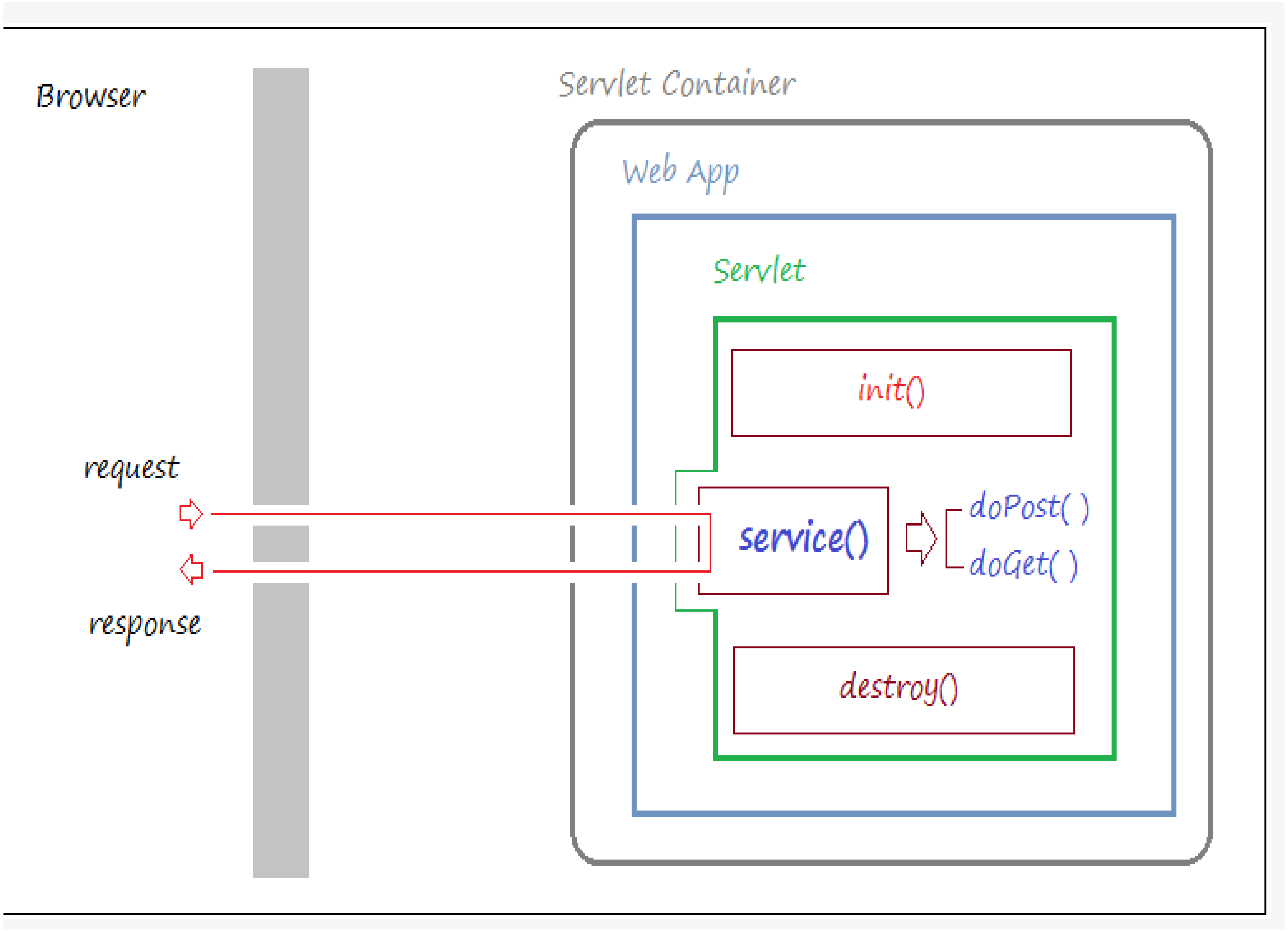


servlet


Client Tier
Web Browser
Middle Tier
Application Server
Database Tier
Database Server

Life Cycle of a Servlet (Servlet Life Cycle) Cont ….




Browser
Servlet Container
Web App
Servlet
init()
request
service()
doPost( )
doGet( )
response
destroy()

1) Servlet class is loaded

- The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

2) Servlet instance is created

- The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

3) init method is invoked

- The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface.
- Syntax of the init method is given below:
- **public void** init(ServletConfig config) **throws** ServletException

4) service method is invoked

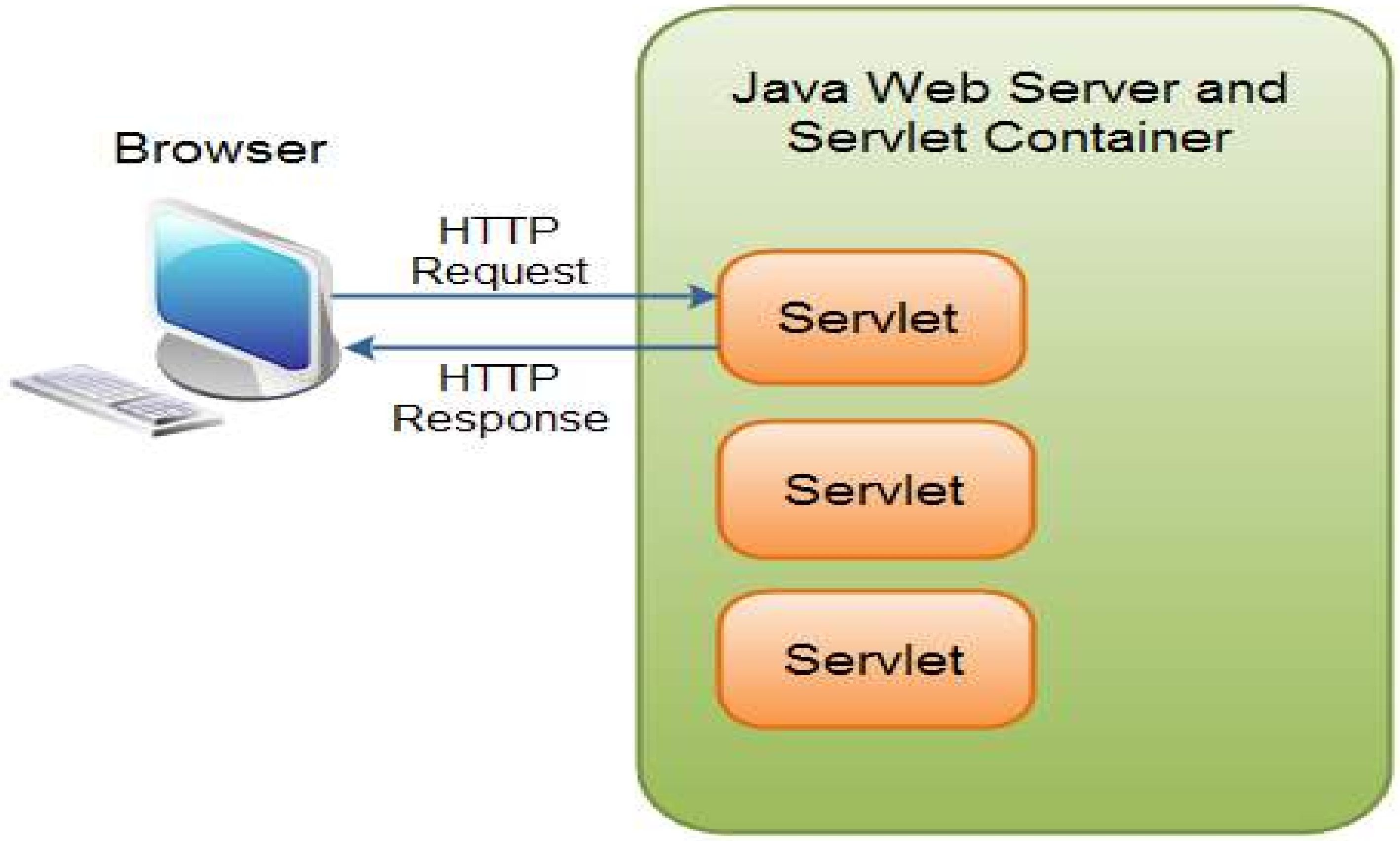
- The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once.
- The syntax of the service method of the Servlet interface is given below:
- **public void** service(ServletRequest request, ServletResponse response) **throws** ServletException, IOException

5) destroy method is invoked

- The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc.
- The syntax of the destroy method of the Servlet interface is given below:
- **public void** destroy()

Servlet Container

- A servlet container is nothing but a compiled, executable program.
- The main function of the container is to load, initialize and execute servlets

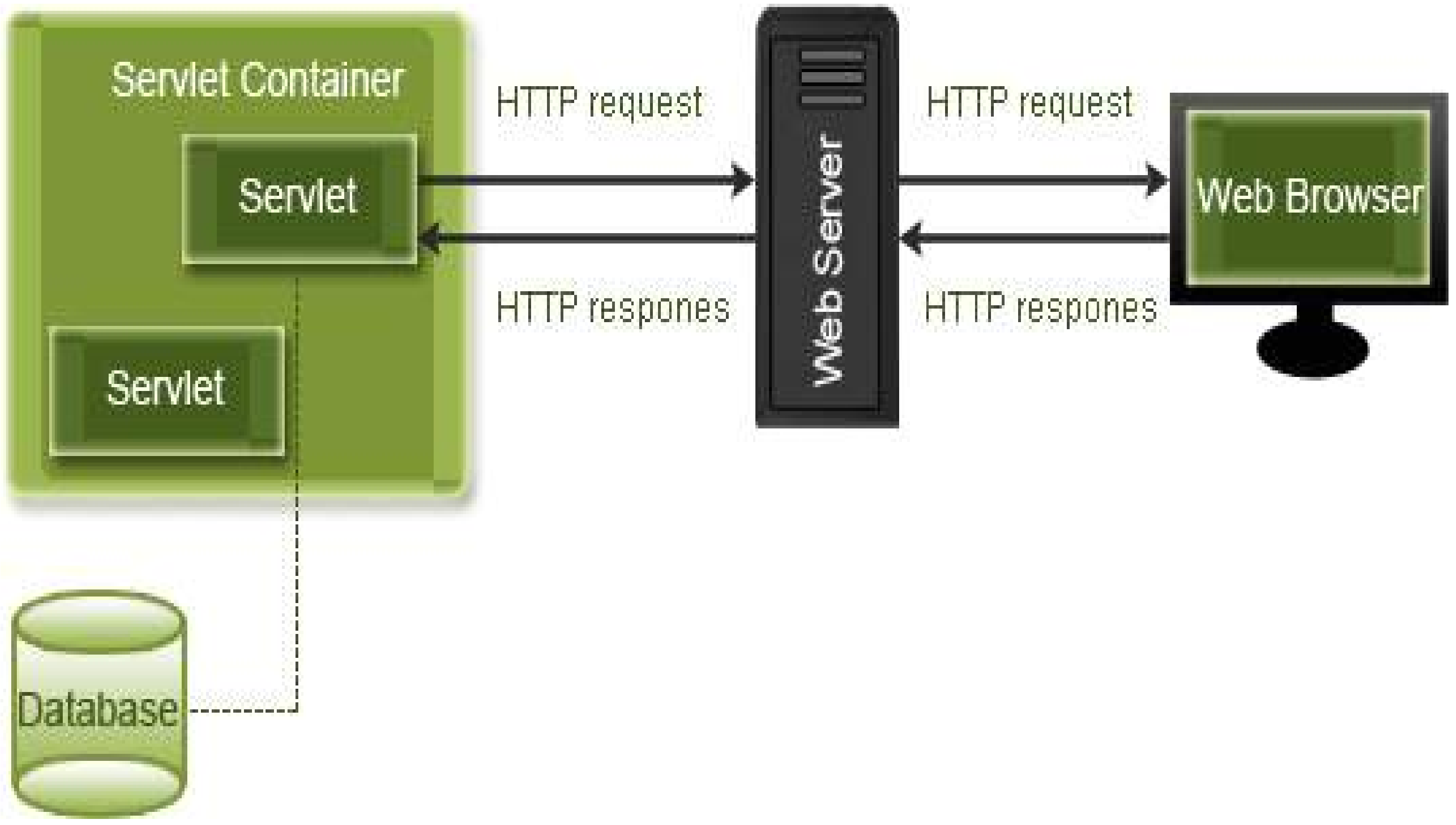


Servlets inside a Java Servlet Container

Servlet Container Cont ….

- A Container have the capacity of containing many servlets and can handle large number of requests. The Container and the Objects of the container both are multi-threaded. It can handle multiple requests concurrently.
- All the servlets are managed by servlet container. It is possible for the servlet container to run stand alone or separately.

Servlet Container



Services provided by the Servlet container :

- **Network Services :** Loads a Servlet class. The loading may be from a local file system, a remote file system or other network services. The Servlet container provides the network services over which the request and response are sent.
- **Decode and Encode MIME based messages :** Provides the service of decoding and encoding MIME-based messages.
- **Manage Servlet container :** Manages the lifecycle of a Servlet.
- **Resource management :** Manages the static and dynamic resource, such as HTML files, Servlets and JSP pages.
- **Security Service :** Handles authorization and authentication of resource access.
- **Session Management :** Maintains a session by appending a **session ID** to the URL path

The Container does the following thing-

- When the request comes at first for the servlet, then it load the servlet, initialize it by calling servlet init() method.
- Makes a thread of the requested servlet
- Makes an instance of ServletRequest and ServletResponse of javax.servlet.*; package
- Passes the ServletRequest and ServletResponse reference variables to the servlet service method
- Take the ServletResponse object from the servlet and passes to the web server
- And finally destroy the reference variables and destroy the servlet when the is shut down.

Servlet API (Application Programming Interface)

- API (Application Programming Interface) is a collection of classes and interfaces. There are mainly two packages in servlet API

Servlets API's:

- Servlets are build from two packages:
- javax.servlet(Basic)
- package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.
- javax.servlet.http(Advance)
- package contains interfaces and classes that are responsible for http requests only.

Various classes and interfaces present in these packages are:

COMPONENT	TYPE	PACKAGE
Servlet	Interface	javax.servlet.*
ServletRequest	Interface	javax.servlet.*
ServletResponse	Interface	javax.servlet.*
GenericServlet	Class	javax.servlet.*
HttpServlet	Class	javax.servlet.http.*
HttpServletRequest	Interface	javax.servlet.http.*
HttpServletResponse	Interface	javax.servlet.http.*
Filter	Interface	javax.servlet.*
ServletConfig	Interface	javax.servlet.*

PACKAGE SUMMARY

- **java.io** package povides the input and output facilities, this package is a standard package of core java platform. It is used for PrintWriter class
- **javax.servlet** :- ServletException class is contained by this package. Every servlet throws the servlet exception therefore it is necessary to include this package in your program.
- **javax.servlet.http** :- HttpServletRequest and HttpServletResponse interfaces are contained by this package. The HttpServletRequest and HttpServletResponse objects created by the servlet container which is passes through the service method doGet(), doPost(), etc. to get servlet request and send servlet response to the client respectively.

Http Servlet -

- HttpServlet is HTTP (Hyper Text Transfer Protocol) specific servlet. It provides an abstract class HttpServlet for the developers for extend to create there own HTTP specific servlets.
- The sub class of HttpServlet must overwrite at least one method given below-
- doGet()
- doPost()
- doTrace()
- doDelete()
- init()
- destroy()
- getServiceInfo()

Difference B/W Generic and HTTP servlet Package

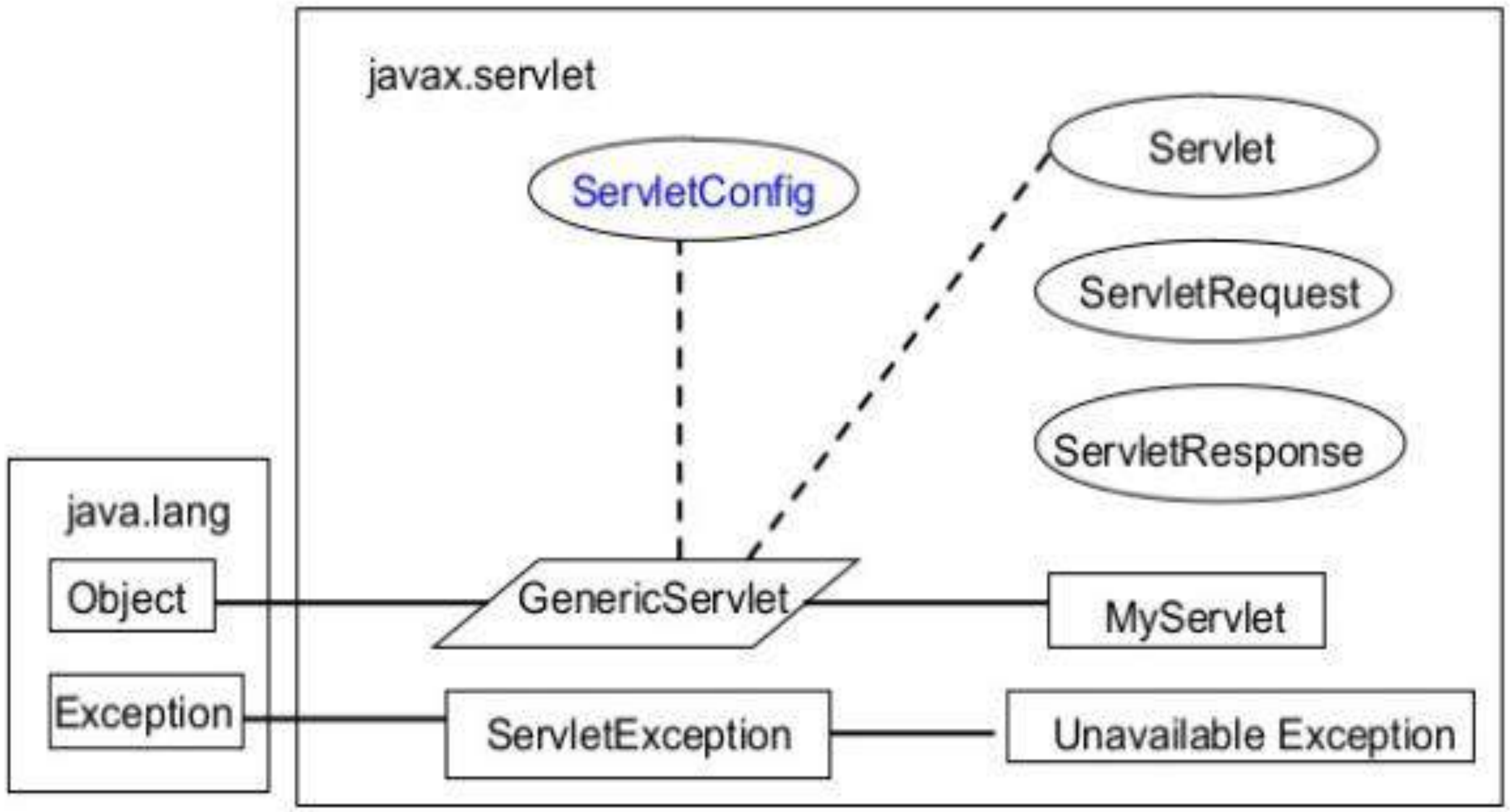
Generic Servlet

- GenericServlet belongs to javax.servlet package while
- **GenericServlet is an abstract class which extends Object and implements Servlet, ServletConfig and java.io.Serializable interfaces**
- . To write a GenericServlet you need abstract service() to be overridden
-)GenericServlet is a protocol-independent servlet

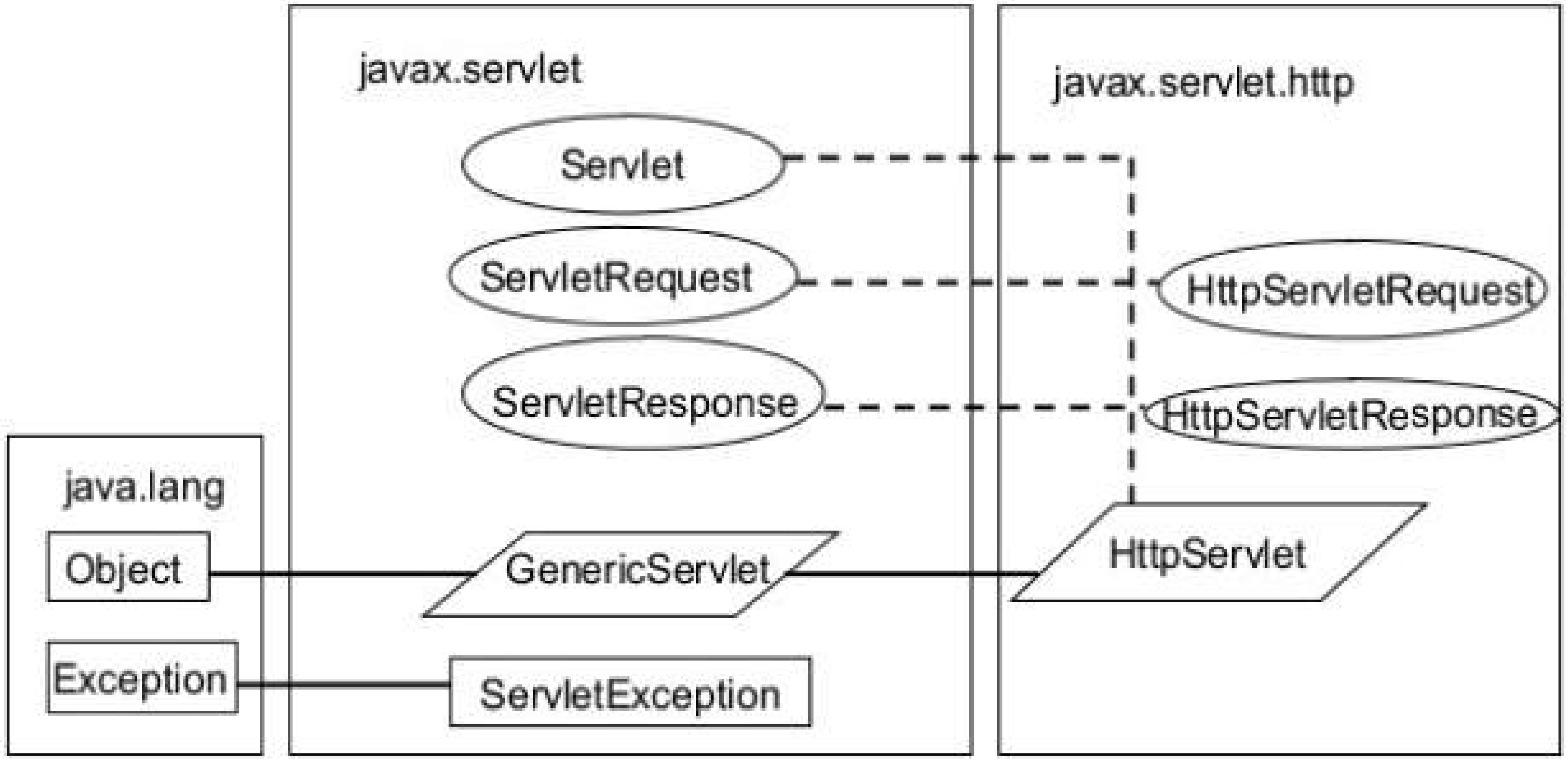
Http Servlet

- HttpServlet belongs to javax.servlet.http package.
- **HTTPServlet is an abstract class which extends GenericServlet and implements java.io.Serializable.**
- HttpServlet must override at least one method of doGet(), doPost(),doPut(), doDelete(), init(), destroy(), getServletInfo().
- HTTPServlet is a protocol dependent servlet.

Javax.servlet package



javax.servlet.http package



javax.servlet.

- This package provides many classes and interfaces, which defines and describes the difference between a servlet class and a run time environment.
- This package provides 14 interfaces and 9 classes.
- They are-

interface

- **Servlet -** This interface defines method for all servlets
- **ServletConfig -** This interface is used by servlet container to pass the parameter to servlet. This interface is used for individual servlet and unknown to the other servlets
- **ServletContext -** This interface contains a set of methods, which is used to communicate with the servlet container and other servlets servlets within an application
- **ServletRequest -** This interface provides a client request information to the servlet
- **ServletResponse -** This interface provides a servlet response information to the client, it assists the servlet to send the response information to the client.
- **RequestDispatcher -** This interface accepts the request from the client sends it to the other application resources.
- **Filter -** It performs the filter task on either request or response to the resources

Interface cont ….

- **FilterChain -** This view a chain of filter request for a resource.
- **FilterConfig -** It used by the servlet container to pass the information during initialization to the of the filter
- **ServletContextListener -** This interface receive a notification when a ServletContext of a web application is initialaised
- **ServletRequestListener -** This interface can be implemented if developer wants to notify when any request out of the scope of web application is coming
- **ServletContextAttributeListener -** This interface receive a notification when a change is made in the ServletContext
- **ServletRequestAttributeListener -** This interface receive a notification when changes is made in request attribute.
- **SingleThreadModel -** This interface is deprecated after the java servlet 2.4.

Classes

- **GenericServlet -** This class defines a generic, protocol independent servlet.
- **ServletContextAttributeEvent -** This is a event class for, when the attribute of an application changes this class receives a nification.
- **ServletContextEvent -** This class receive a notifications when the ServletContext of an application is changes.
- **ServletInputStream -** This provides a servlet input stream for reading binary data from the client.
- **ServletOutputStream -** This class provides an output steam for sending binary data to the client.
- **ServletRequestWrapper -** This class provides the implementation of ServletRequest interface, the developers can subclass this for their own specific use
- **ServletResponseWrapper -** This class provides the implementation of ServletResponse interface, the developers can subclass this for their own specific use
- **ServletRequestAttributeEvent -** This is a event class it receive a notification when any change made in the request attribute of the application
- **ServletRequestEvent -** This event class indicates the events of the servlet class

javax.servlet.http.

- This package provides classes and interfaces which is HTTP protocol specific.
- There are 8 interfaces and 7 classes in this package

Interfaces

- **HttpServletRequest -** This interface extends the ServletRequest interface for provide http specific servlet request
- **HttpServletResponse -** This interface extends the ServletResponse interface for provide http specific servlet response
- **HttpSession -** It provides a way for identifying a user on the website
- **HttpSessionActivationListener -** This interface notify when when session is Activated
- **HttpSessionAttributeListener -** This interface can be implemented in order to get notification when attribute of the session is changed
- **HttpSessionBindingListener -** This interface can be implemented in order to get notification when object is bound or unbound to the session
- **HttpSessionContext -** This interface is deprecated in servlet 2.1 for security resion
- **HttpSessionListener -** When developer want to notify the session activation or de-activation, can implement this interface

Classes

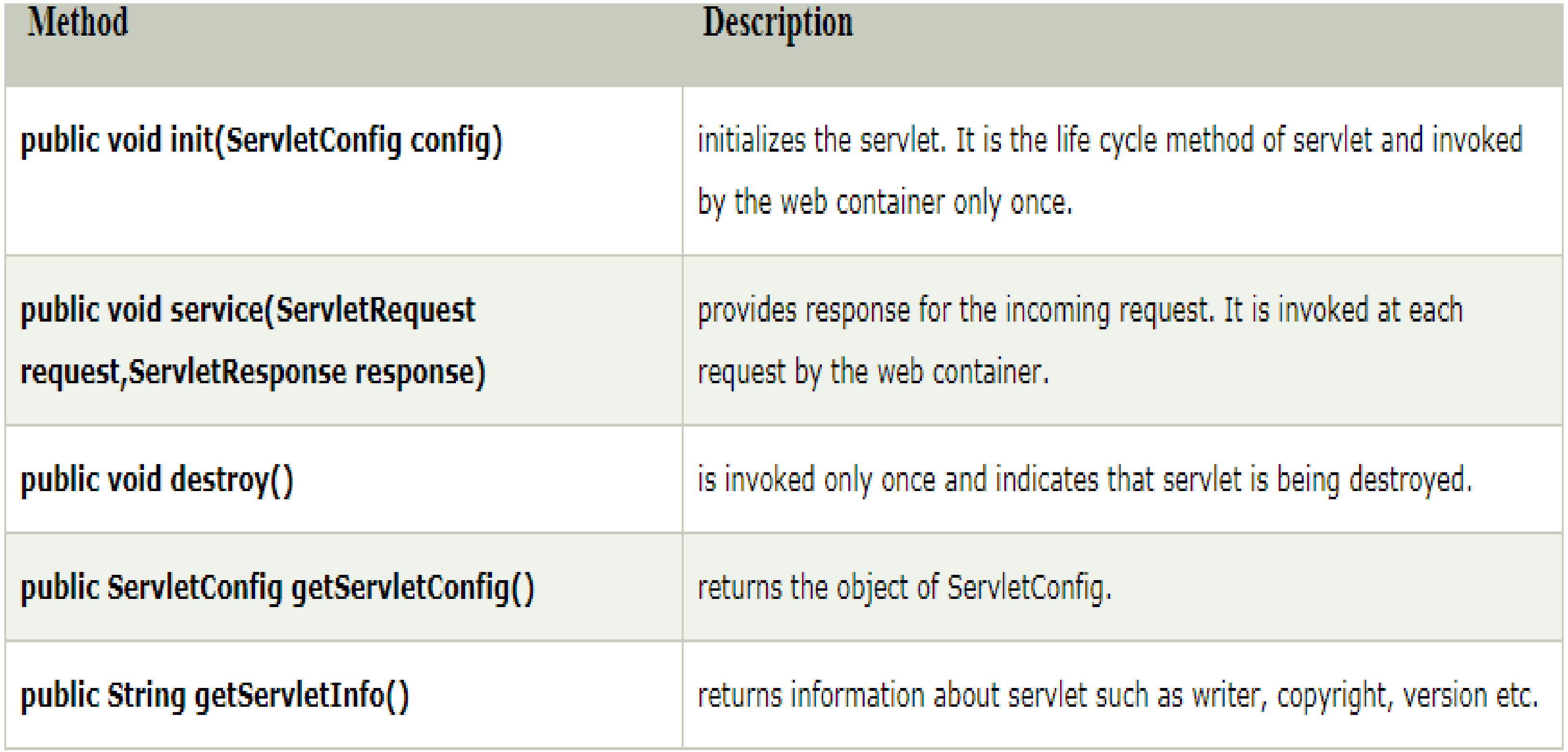
- **Cookie -** It creates a cookie sent by a servlet to the web browser, which browser saves and latter return for further communication
- **HttpServlet -** This is an abstract class, to create a Http specific servlet you need to extend this class
- **HttpServletRequestWrapper -**This class implements the HttpServletRequest interface, the developer can also subclass this for their specific use
- **HttpServletResponseWrapper -** This class implements the HttpServletResponse interface, the developer can also subclass this for their specific use
- **HttpSessionBindingEvent -** This is an events class, it is activated when session is bound and unbound
- **HttpSessionEvent -** This event class generates a notification for changes to the session within a web application
- **HttpUtils -** This class is deprecated in servlet 2.3 API

Servlet Interface

- **Servlet interface provides** common behavior to all the servlets.Servlet interface defines methods that all servlets must implement.
- Servlet interface needs to be implemented for creating any servlet (either directly or indirectly). It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

Methods of Servlet interface

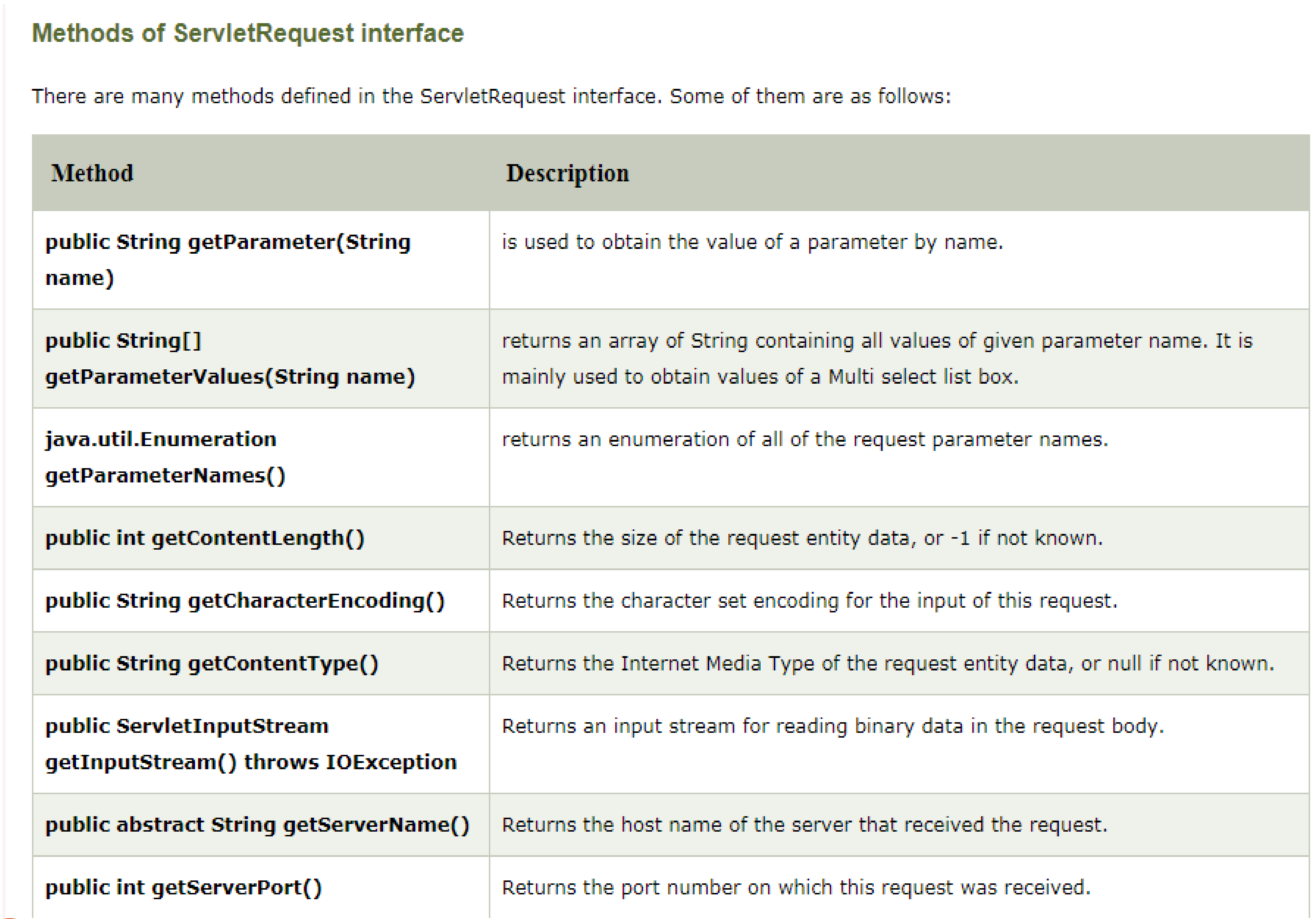
There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.



Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

ServletRequest Interface

- An object of ServletRequest is used to provide the client request information to a servlet such as content type, content length, parameter names and values, header informations, attributes etc



Methods of ServletRequest interface

There are many methods defined in the ServletRequest interface. Some of them are as follows:

Method	Description
public String getParameter(String name)	is used to obtain the value of a parameter by name.
public String[] getParameterValues(String name)	returns an array of String containing all values of given parameter name. It is mainly used to obtain values of a Multi select list box.
java.util.Enumeration getParameterNames()	returns an enumeration of all of the request parameter names.
public int getContentLength()	Returns the size of the request entity data, or -1 if not known.
public String getCharacterEncoding()	Returns the character set encoding for the input of this request.
public String getContentType()	Returns the Internet Media Type of the request entity data, or null if not known.
public ServletInputStream getInputStream() throws IOException	Returns an input stream for reading binary data in the request body.
public abstract String getServerName()	Returns the host name of the server that received the request.
public int getServerPort()	Returns the port number on which this request was received.

Index.html

- <body>
- <form action ="first">
-             <p> Name<input type="text" name="username"></p>
-             <p> <input type="submit" value="first"></p>
-         </form>
- </body>

SERVLET CODE

- out.println("<h1>Welcome to my first servlet</h1>");
- String user = request.getParameter("username");
- out.println("<h2> Welcome "+user+"</h2>");
- java.util.Date date = new java.util.Date();
- out.println("<h2>"+"Current Date & Time: " +date.toString()+"</h2>");
-

QUESTION:
DISPLAY DATAE AND TIME IN THIS FORMAT

- Display Current Date & Time
- Mon 2010.06.21 at 10:06:44 PM GMT+04:00

Html code

- **<form action="Counter">**
- <h1> <input type="submit" value="submit"></h1>
- </form>

HIT COUNTER

Servlet interface provides common behavior to all the servlets.Servlet interface defines methods that all servlets must implement.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Counter extends HttpServlet{
  int counter = 0;

  public void doGet(HttpServletRequest request, HttpServletResponse
    response)throws ServletException, IOException {

  response.setContentType("text/html");

  PrintWriter pw = response.getWriter();
  counter++;
  pw.println("TOTAL COUNT= " + counter);
  }
}
```

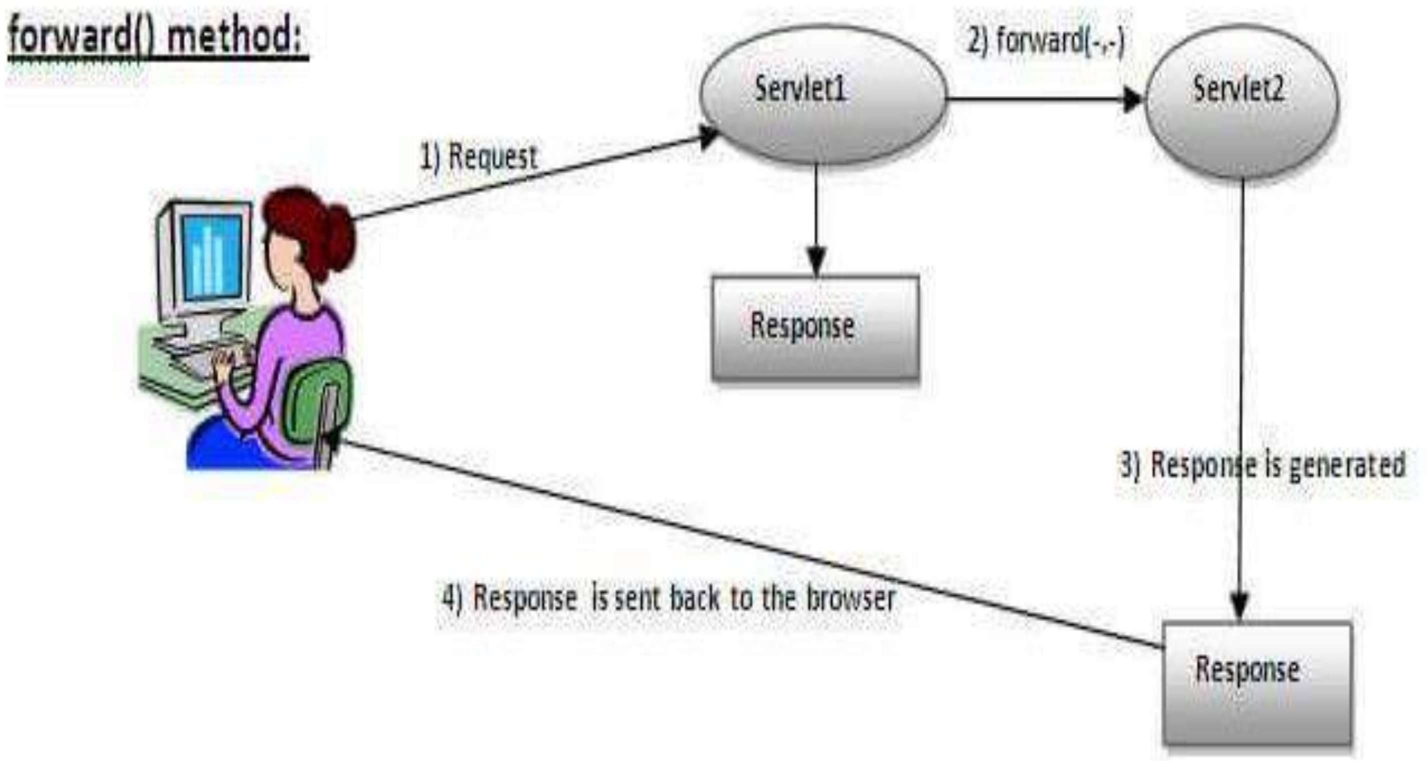
RequestDispatcher in Servlet

- The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp.
- This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.
- There are two methods defined in the RequestDispatcher interface.

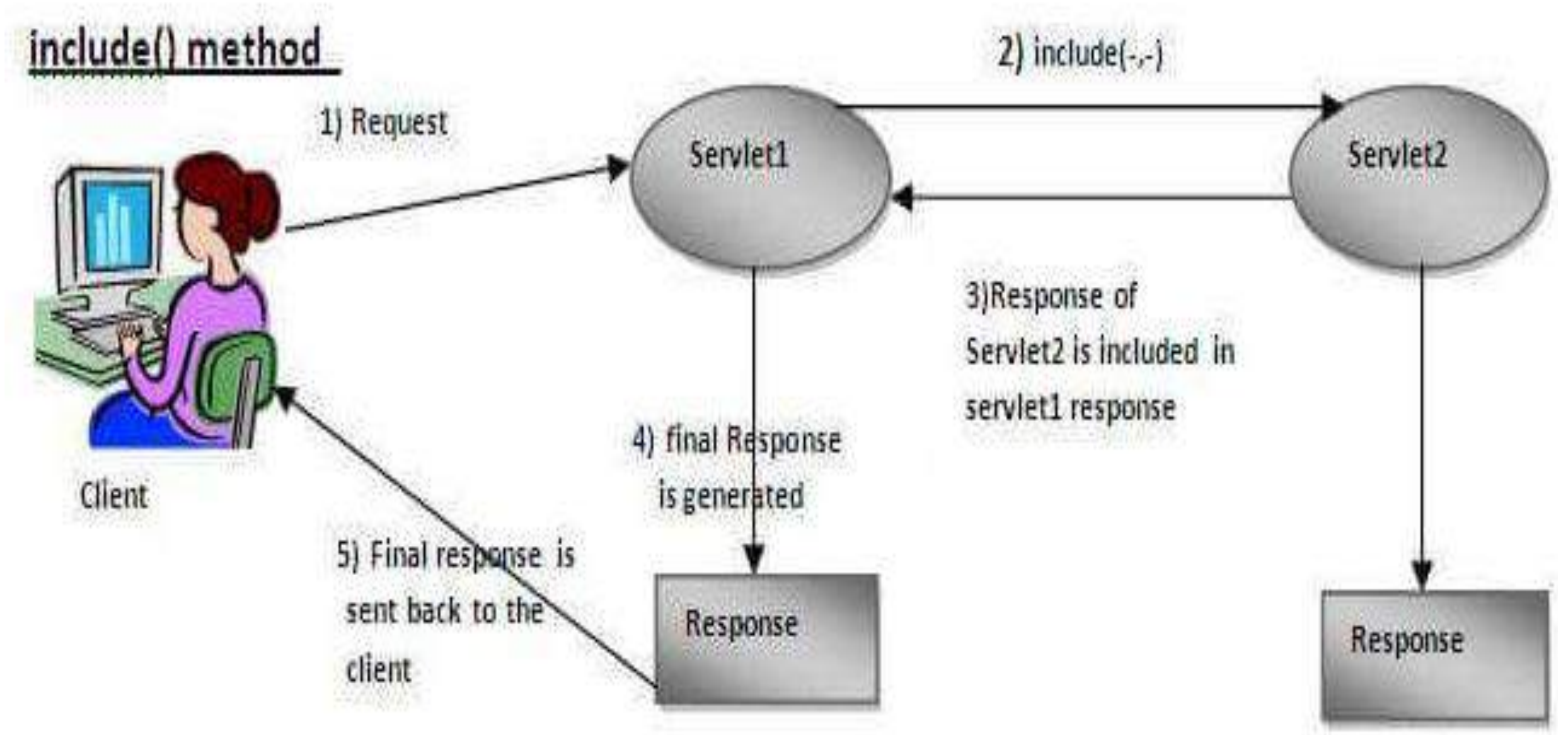
Methods of RequestDispatcher interface

- The RequestDispatcher interface provides two methods. They are:
- **public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**
- Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
- **public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Includes
- the content of a resource (servlet, JSP page, or HTML file) in the response.

IN GIVEN figure, response of second servlet is sent to the client. Response of the first servlet is not displayed to the user.



IN GIVEN figure, response of second servlet is included in the response of the first servlet that is being sent to the client.


include() method
1) Request
2) include(-,-)
Servlet1
Servlet2
3)Response of Servlet2 is included in servlet1 response
4) final Response is generated
Client
5) Final response is sent back to the client
Response
Response

FORWARD()	SENDREDIRECT()
The forward() method is executed in the server side.	The sendRedirect() method is executed in the client side.
The request is transfer to other resource within same server.	The request is transfer to other resource to different server.
It does not depend on the client's request protocol since the forward ( ) method is provided by the servlet container.	The sendRedirect() method is provided under HTTP so it can be used only with HTTP clients.
The request is shared by the target resource.	New request is created for the destination resource.
Only one call is consumed in this method.	Two request and response calls are consumed.
It is declared in RequestDispatcher interface.	It is declared in HttpServletResponse.

FORWARD()	SENDREDIRECT()
The forward() method is faster than sendRedirect() method.	The sendRedirect() method is slower because when new request is created old request object is lost.
It can be used within server.	It can be used within and outside the server.
Signature : *forward(ServletRequest request, ServletResponse response)*	Signature: *void sendRedirect(String url)*

Some more points to notice of forward vs sendRedirect.

- Session is not lost in both cases.
- The above differences are applicable to Servlets and JSPs. In Servlets, these methods are used in service() and in JSP used in scriptlets.
- In frameworks like Struts, the Controller can decide, at the end of request processing, which one to use of either forward or redirect operation.
- The Controller also can decide with forward() method, to what resource the forward should be made, depending on different conditions of client request requirements.

Which one is preferred?

- Just depends on the scenario.
- If you would like to forward the client request to a new resource on the same server for further process, prefer forward() where data of the original resource can be passed to the new resource.
- If you would like to transfer the control to a new server or domain where client treats as a new task, prefer sendRedirect(). If the data of the original resource (which client requested) is needed in the new resource, store them in Session object and reuse.

Html code

```
 <body>
      <form action="Login" method="post">
Name:<input type="text" name="userName"/><br/>
Password:<input type="password"
   name="userPass"/><br/>
<input type="submit" value="login"/>
</form>
    </body>
```

Login.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
   public class Login extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse response)
       throws ServletException, IOException {
     response.setContentType("text/html");
   PrintWriter out = response.getWriter();

   String username=request.getParameter("userName");
   String paas=request.getParameter("userPass");

   if(paas.equals("sharma"))
   {
    RequestDispatcher rd=request.getRequestDispatcher("welcomeservlet");
      rd.forward(request, response);
   }
   else{
      out.print("Sorry UserName or Password Error!");
      RequestDispatcher rd=request.getRequestDispatcher("/index.html");
      rd.include(request, response);

      }      }      }
```

welcomeservlet

- out.println("<h1>welcome to servlet </h1>");

- //String user = request.getParameter("username");
- //	out.println("<h2> Welcome "+user+"</h2>");

SendRedirect in servlet

- The **sendRedirect()** method of **HttpServletResponse** interface can be used to redirect response to another resource, it may be servlet, jsp or html file.
- It accepts relative as well as absolute URL.
- It works at client side because it uses the url bar of the browser to make another request. So, it can work inside and outside the serve

HTML CODE

- <form action=" SendServlet ">
- <h1> <input type="submit" value="submit"></h1>
- </form>

sendRedirect()

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SendServlet extends HttpServlet{
public void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException
{
res.setContentType("text/html");
PrintWriter pw=res.getWriter();

response.sendRedirect("https://www.amazon.in/");

pw.close();
}}
```

HTML CODE

- **<form** action=“GoogleSearch">
- **<input** type="text" name="name">
- **<input** type="submit" value="Google Search">
- **</form>**

sendRedirect()

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

 public class  GoogleSearch  extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletRespo
  nse response)   throws ServletException, IOException {

      String name=request.getParameter("name");
      response.sendRedirect("https://www.google.co.in/#q="+name);
  }
}
```

HTML FILE FOR Multiple values for a single parameter

```
<form action="GetParameterValues" method="post">

    BAALUSHAH <input type="checkbox" name="Sweetdish"
 value="BAALUSHAHI" > </input>

  BARFI     <input type="checkbox" name="Sweetdish"
 value="BARFI" /> </input>

     <input type="checkbox" name="Sweetdish"
 value="RASGULLA" />RASGULLA </input>
     <input type="submit">

     </form>
```

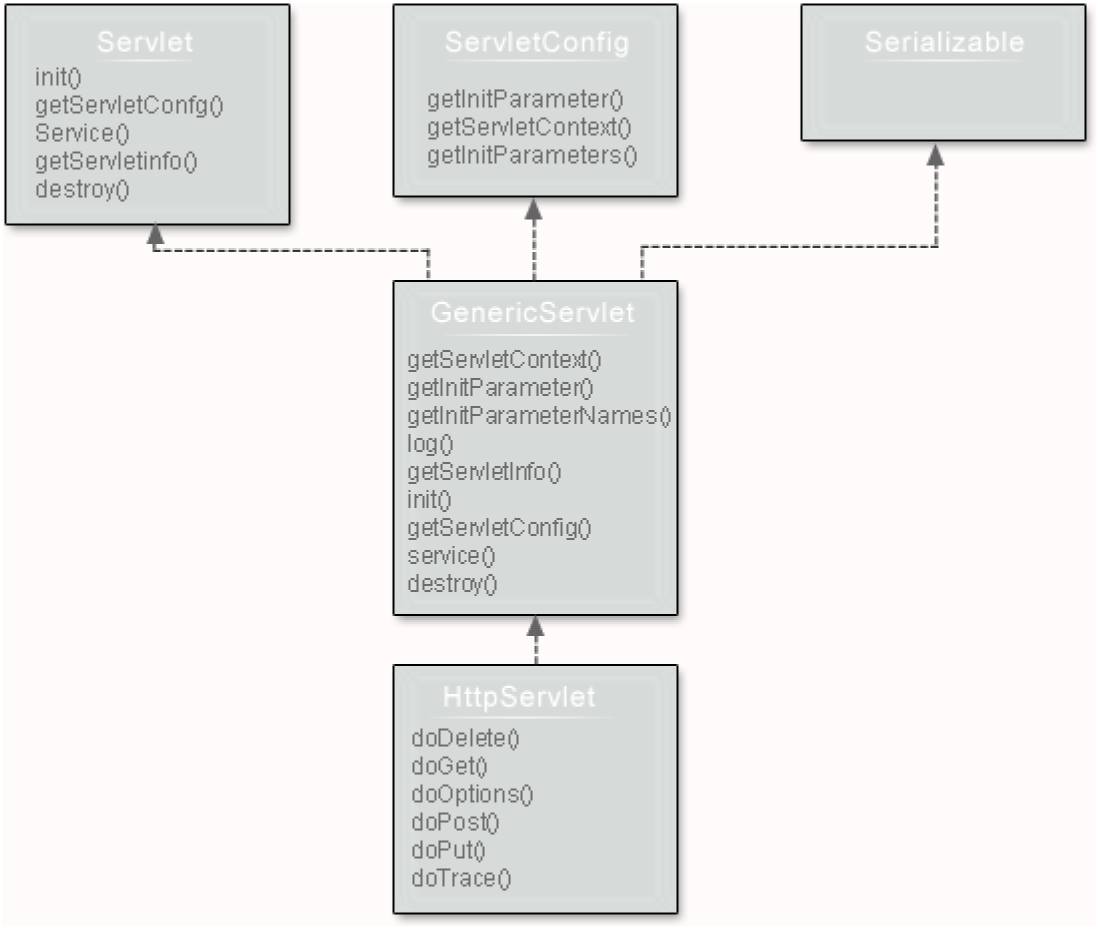
Reading Parameter values

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class GetParameterValues extends HttpServlet{

  protected void doPost(HttpServletRequest request,
   HttpServletResponse response)
 throws ServletException, IOException
  {
  response.setContentType("text/html");
  PrintWriter pw = response.getWriter();
  String[]  Sweetdish  = request.getParameterValues("Sweetdish");
  for(int i=0; i< Sweetdish.length; i++){
  pw.println("<br> Sweetdish  : " +  Sweetdish[i]);
  }   } }
```

Servlet types

There are mainly two types of servlets -

Generic Servlet - Generic servlet is protocol independent servlet. It implements the Servlet and ServletConfig interface. It may be directly extended by the servlet. Writing a servlet in in GenericServlet is very easy. It has only init() and destroy() method of ServletConfig interface in its life cycle. It also implements the log method of ServletContext interfac



ServletConfig Interface

- An object of ServletConfig is created by the web container for each servlet. This object can be used to get configuration information from web.xml file.
- If the configuration information is modified from the web.xml file, we don't need to change the servlet. So it is easier to manage the web application if any specific content is modified from time to time.

Advantage of ServletConfig

- The core advantage of ServletConfig is that you don't need to edit the servlet file if information is modified from the web.xml file.

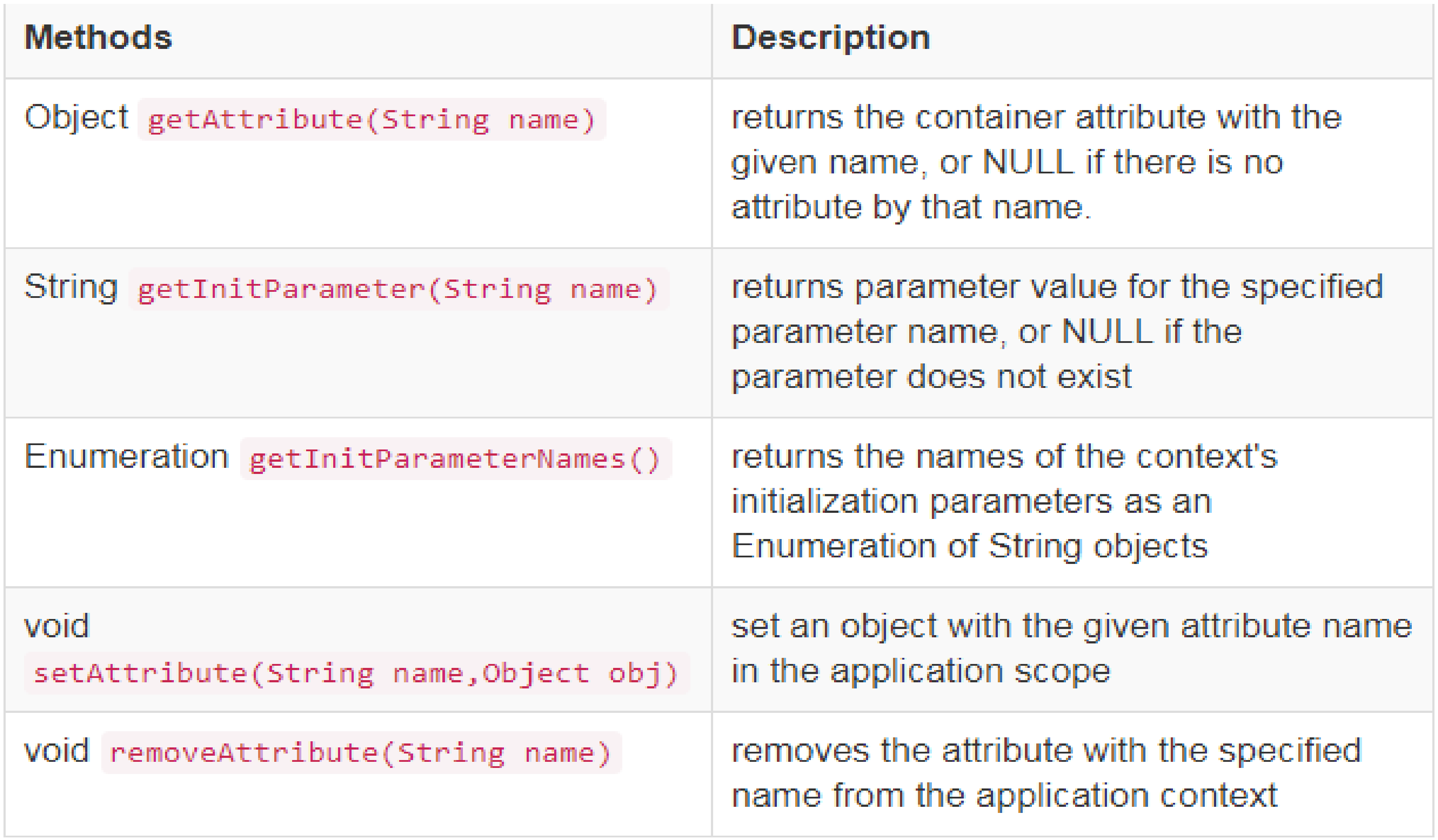
How to get the object of ServletConfig

- **getServletConfig() method** of Servlet interface returns the object of ServletConfig.
- Syntax of getServletConfig() method
- **public** ServletConfig getServletConfig();

ServletContext Interface in Servlet

- For every **Web application** a **ServletContext** object is created by the web container. ServletContext object is used to get configuration information from **Deployment Descriptor**(web.xml) which will be available to any servlet or JSPs that are part of the web app.

Commonly used methods of ServletContext interface



Methods	Description
Object `getAttribute(String name)`	returns the container attribute with the given name, or NULL if there is no attribute by that name.
String `getInitParameter(String name)`	returns parameter value for the specified parameter name, or NULL if the parameter does not exist
Enumeration `getInitParameterNames()`	returns the names of the context's initialization parameters as an Enumeration of String objects
void `setAttribute(String name,Object obj)`	set an object with the given attribute name in the application scope
void `removeAttribute(String name)`	removes the attribute with the specified name from the application context

How to get the object of ServletContext interface

- getServletContext() method of ServletConfig interface returns the object of ServletContext.
- getServletContext() method of GenericServlet class returns the object of ServletContext.
- Syntax of getServletContext() method
- public ServletContext getServletContext()

Advantages of ServletContext

- Provides communication between servlets
- Available to all servlets and JSPs that are part of the web app
- Used to get configuration information from web.xml

Difference between Context Init Parameters and Servlet Init Parameter

Context Init parameters	**Servlet Init parameter**
Available to all servlets and JSPs that are part of web	Available to only servlet for which the <init-param> was configured
Context Init parameters are initialized within the `<web-app>` not within a specific `<servlet>` elements	Initialized within the `<servlet>` for each specific servlet.
ServletContext object is used to get Context Init parameters	ServletConfig object is used to get Servlet Init parameters
Only one ServletContext object for entire web app	Each servlet has its own ServletConfig object

HTML FILE

```
<form action="MyServlet">
        <input type="submit" value="submit">
    </form>
```

SERVLET FILE

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {

 protected void doGet(HttpServletRequest request, HttpServletResponse response)   throws
    ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

 PrintWriter out = response.getWriter();
        ServletContext sc = getServletContext();

        out.println(sc.getInitParameter("dname"));

   }}
```

Web.xml

```
<context-param>
<param-name>dname</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-
   value>
</context-param>
```

ServletConfig

ServletConfig available in javax.servlet.*; package

ServletConfig object is one per servlet class

Object of ServletConfig will be created during initialization process of the servlet

This Config object is public to a particular servlet only

Scope: As long as a servlet is executing, ServletConfig object will be available, it will be destroyed once the servlet execution is completed.

We should give request explicitly, in order to create ServletConfig object for the first time

In web.xml – <*init-param*> tag will be appear under <*servlet-class*> tag

ServletContext

ServletContext available in javax.servlet.*; package

ServletContext object is global to entire web application

Object of ServletContext will be created at the time of web application deployment

Scope: As long as web application is executing, ServletContext object will be available, and it will be destroyed once the application is removed from the server.

ServletContext object will be available even before giving the first request

In web.xml – <*context-param*> tag will be appear under <*web-app*> tag

So finally…….

No. of web applications = That many number of ServletContext objects [1 per web application]
No. of servlet classes = That many number of ServletConfig objects

Difference between ServletConfig and ServletContext

- The servletconfig object refers to the single servlet whereas servletcontext object refers to the whole web application.

Attribute in Servlet

- An **attribute** is an object that is used to share information in a web app. Attribute allows Servlets to share information among themselves.
- Attributes can be SET and GET t or removed from one of the following scopes

Attribute in Servlet

- 1.**Request:** The request attribute can be accessible within that same servlet Request.
- 2.**Session:** these attributes are accessible within the same Http Session.
- 3.**Context /Application:** any servlet within the same application can have access to context attributes.
- The servlet programmer can pass information from one servlet to another using attributes. It is just like passing object from one class to another so that we can reuse the same object again and again.

What we do with these attributes?

- setAttribute (): This method is used to set the attribute values into objects.
- getAttribute (): This method is used to get the attribute values from scope objects.
- getAttributeNames (): This method is used to get the name of Attribute objects.
- removeAttribute (): this method is used to remove attributes from the respective scopes of objects.

Attribute specific methods of ServletRequest, HttpSession and ServletContext interface

- public void setAttribute(String name,Object object):sets the given object in the application scope.
- public Object getAttribute(String name):Returns the attribute for the specified name.
- public Enumeration getInitParameterNames():Returns the names of the context's initialization parameters as an Enumeration of String objects.
- public void removeAttribute(String name):Removes the attribute with the given name from the servlet context.

HTML FILE

```
<form action="First">
        <input type="submit" value="submit">
    </form>
```

First.java to SET an Attribute

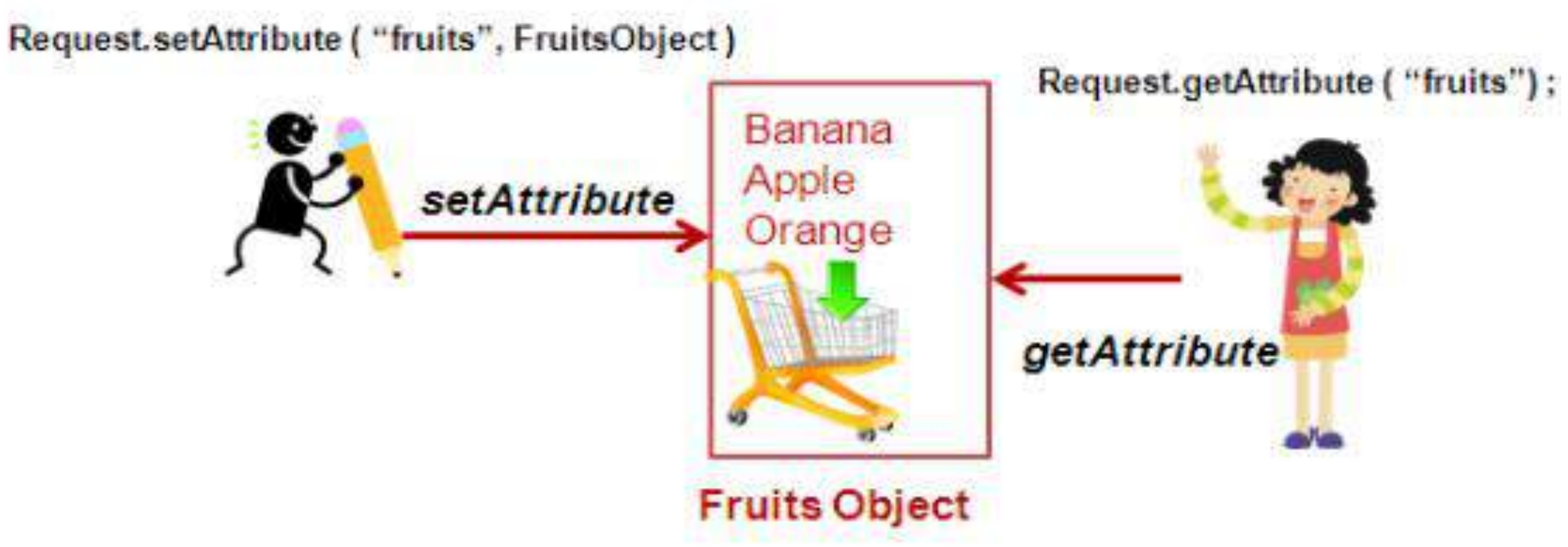
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class First extends HttpServlet {

  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws
     ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        ServletContext sc = getServletContext();
        sc.setAttribute("user","MCA");            //setting attribute on context scope

        out.println("<a href='Second'>visit</a>");
    }
}
```

Second.java to GET An Attribute

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Second extends  HttpServlet {

 protected void  doGet(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      ServletContext sc = getServletContext();

      String str = (String) sc.getAttribute("user");  //getting attribute from context scope

      out.println("Welcome"+str);  // Prints : Welcome  MCA
   }
}
```

How servlet is used with each scope of Object?

- **1.Request Attribute:**
- “This object attributes are accessible only within that same servlet request object.”
- We can set and get Request Attributes with an object.
  You can do all the operations mentioned above with request object.
  For example:
- Request.setAttribute(“fruits”,FruitsObject);

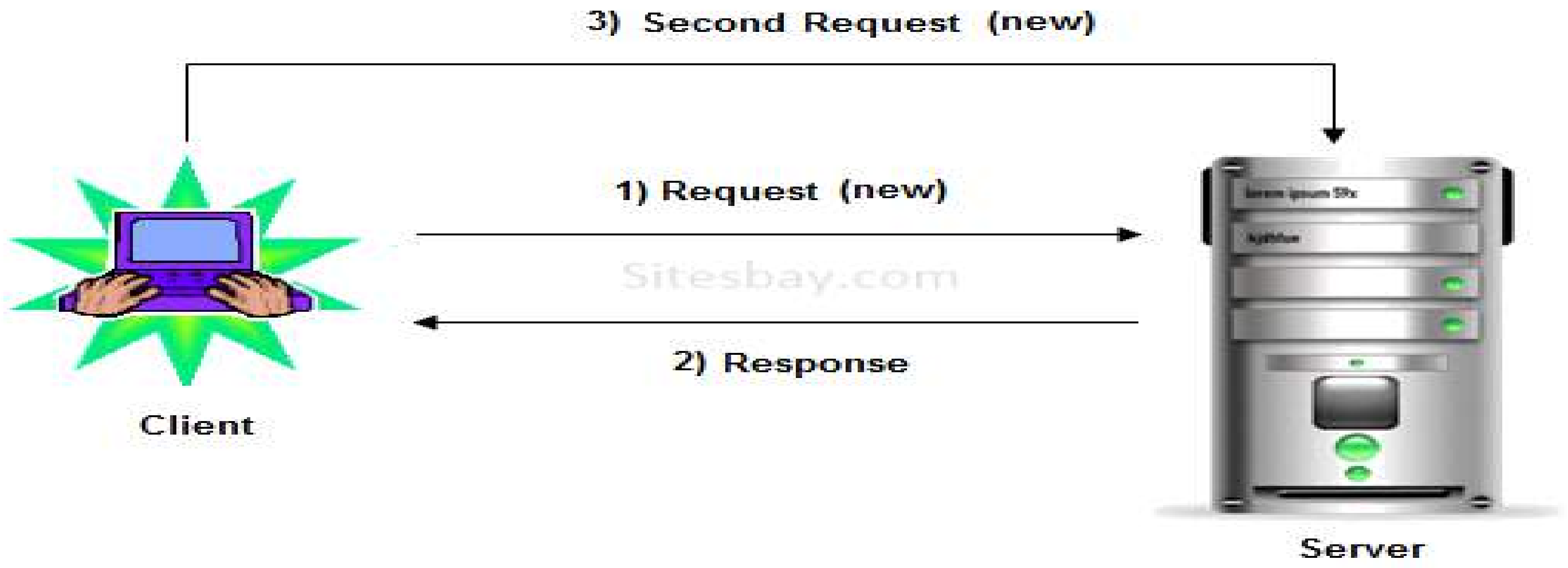

Request.setAttribute ("fruits", FruitsObject)
Request.getAttribute ("fruits") ;
setAttribute
Banana
Apple
Orange
getAttribute
Fruits Object

Session Tracking in Servlet

- **Session** is the conversion of user within span of time. In general meaning particular interval of time.
- **Tracking** is the recording of the thing under session.
- **Session Tracking is remembering and recording of client conversion in span of time. It is also called as session management.**
- If web application is capable of remembering and recording of client conversion in span of time then that web application is called as **stateful web application**.

Why need Session Tracking ?

- Http protocol is stateless, to make stateful between client and server we need Session Tracking.
- Session Tracking is useful for online shopping, mailing application, E-Commerce application to track the conversion.
- Http protocol is stateless, that means each request is considered as the new request. You can see in below image



Why use Session Tracking ?

To recognize the user It is used to recognize the particular user.

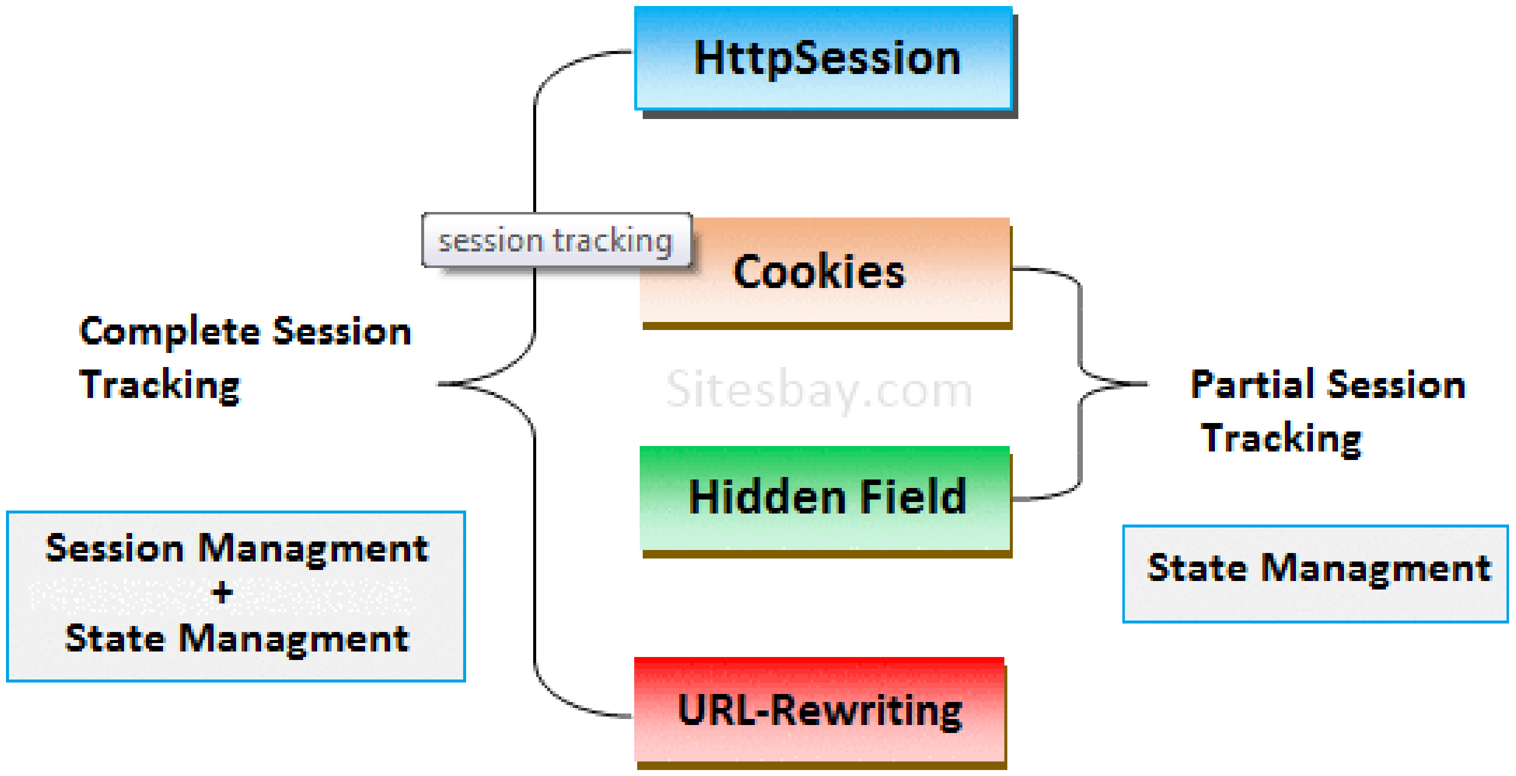
Why Http is design as stateless protocol ?

If Http is stateful protocol for multiple requests given by client to web application single connection will be used between browser and web server across the multiple requests. This may make clients to engage connection with web server for long time event though the connection are ideal. Due to this the web server reach to maximum connections even though most of its connection are idle. To overcome this problem Http is given as stateles

Session tracking methods:

- User authorization
- Hidden fields
- URL rewriting
- Cookies
- Session tracking API

SESSION TRACKING TECHNIQUES



BUILD ON THE TOP OF ALL ABOVE 4 APIS'

SESSION API

Multiple CheckBox Example

-
  ```
  import java.io.*;
  import javax.servlet.*;
  import javax.servlet.http.*;
  ```

-
  ```
  public class CheckBox extends HttpServlet{
  public void doGet(HttpServletRequest req, HttpServletResponse res)throws ServletException, IOException{
  String name;
  String[] hobbies;
  res.setContentType("text/html");
  name= req.getParameter("name");
  PrintWriter pw = res.getWriter();
  hobbies= req.getParameterValues("hobbies");
  if(hobbies!=null)
  {
  pw.println("<html><body bgcolor=#00ffff>");
  pw.println("Hi I am <h4>"+name+".</h4><br>");
  pw.println("My hobby/hobbies are:");
  for(int i=0; i<hobbies.length; i++){
  pw.println(hobbies[i]+",");
  }
  pw.println("</body></html>");
  }
  else
  pw.println("<p><font color=red>not selected</font></p>");
  }
  }
  ```

Deployment Descriptors

- A web application's deployment descriptor describes the classes, resources and configuration of the application and how the web server uses them to serve web requests. When the web server receives a request for the application, it uses the deployment descriptor to map the URL of the request to the code that ought to handle the request.
- The deployment descriptor is a file named web.xml. It resides in the app's WAR under the WEB-INF/ directory. The file is an XML file whose root element is <web-app>.

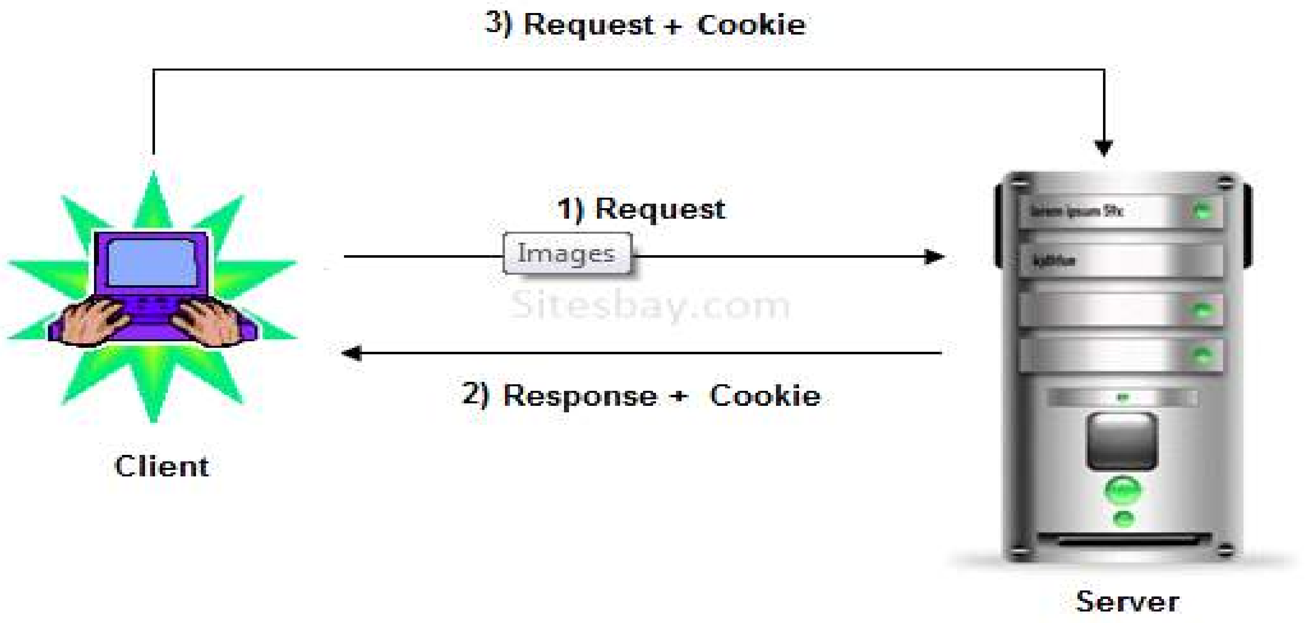
Cookie

cookie

- **Cookies** are text files stored on the client computer and they are kept for various information like name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.
  - **or**
- A **cookie** is a small piece of information that is persisted between the multiple client requests.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

How Cookie works

- By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



Cookies in Servlet

- A **cookie** is a small piece of information that is persisted between the multiple client requests.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number

How Cookie works

- By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old use

When use cookies ?

- When session ID is not required and when less number of input values are submitted by client in that case in place of using HttpSession Technique you can use cookies Technique to reduce the burden on server.

Points to Remember

- Cookies is pressistance resource which is stores at client location.
- We can store 3000 cookies in cookies file at a time.
- The cookies are introduced by net scape communication.
- Cookies files exist up to 3 year.
- Size of cookies is 4 kb.

Types of Cookie

- There are 2 types of cookies in servlets.
  1. Non-persistent cookie
  2. Persistent cookie

- Non-persistent cookie
- It is **valid for single session** only. It is removed each time when user closes the browser.

- Persistent cookie
- It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

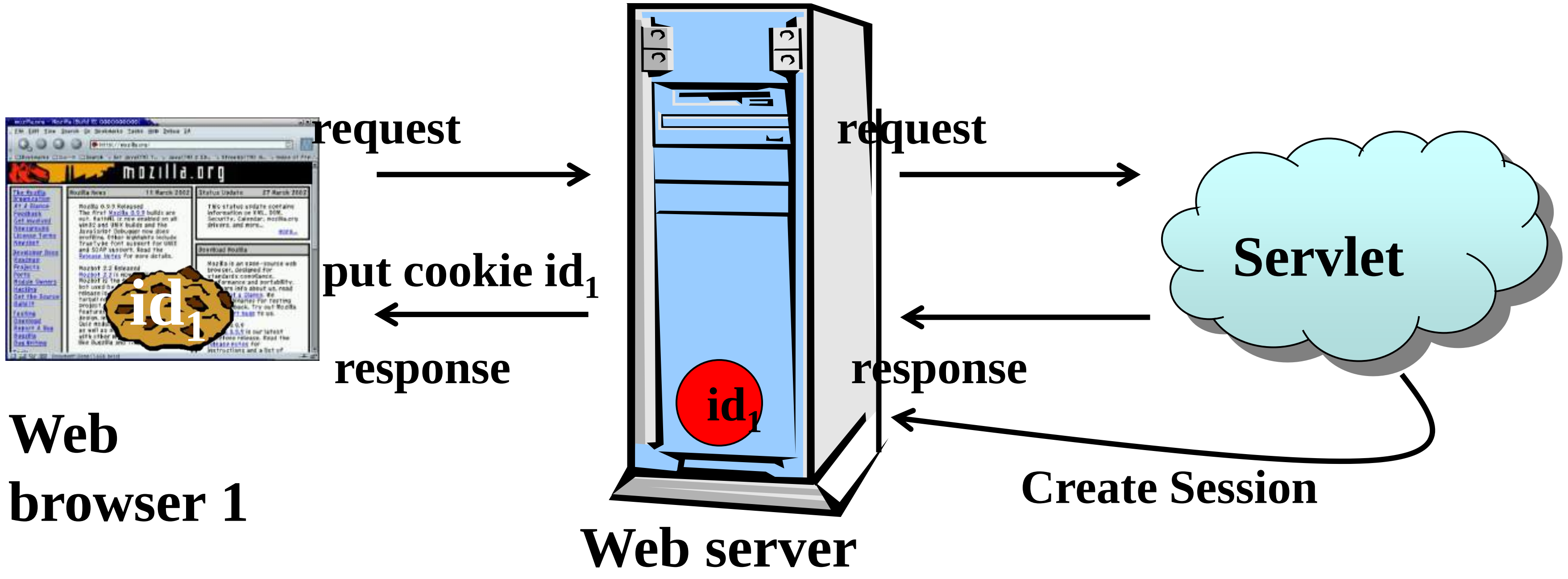
Advantage of Cookies

- Simplest technique of maintaining the state.
- Cookie are maintained at client side so they do not give any burden to server.
- All server side technology and all web server, application servers support cookies.
- Presistent cookies can remember client data during session and after session with expiry time.

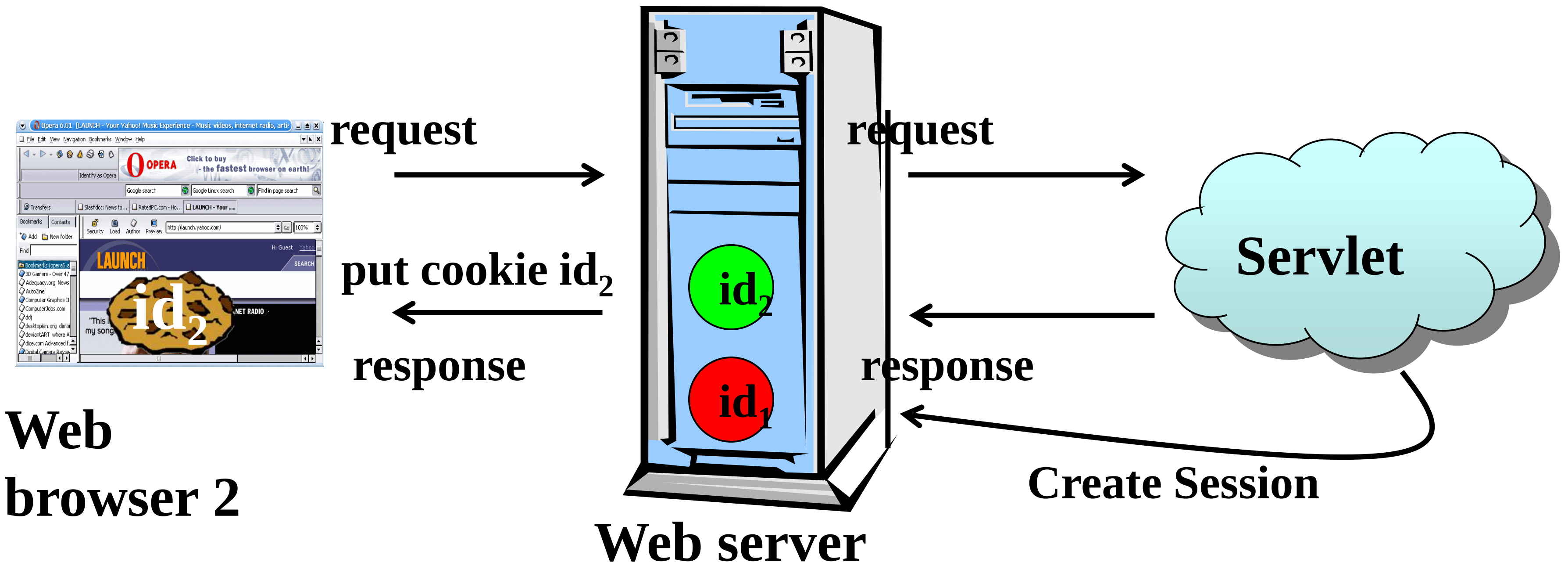
Limitation of Cookies

- It will not work if cookie is disabled from the browser.
- Cookies are text files, It does not provides security. Any one can change this file.
- With coockies need client support that means if client disable the coockies then it does not store the client location.
- Cookies can not store java objects as values, they only store text or string.

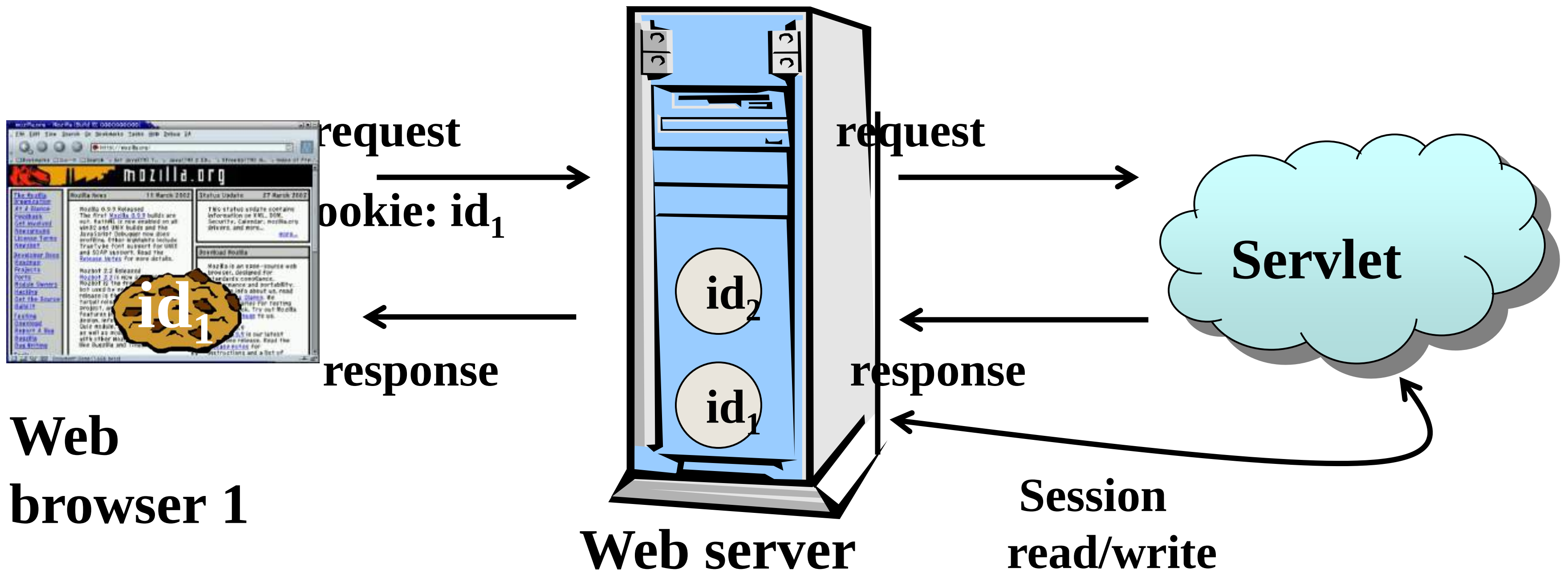
Session Cookies



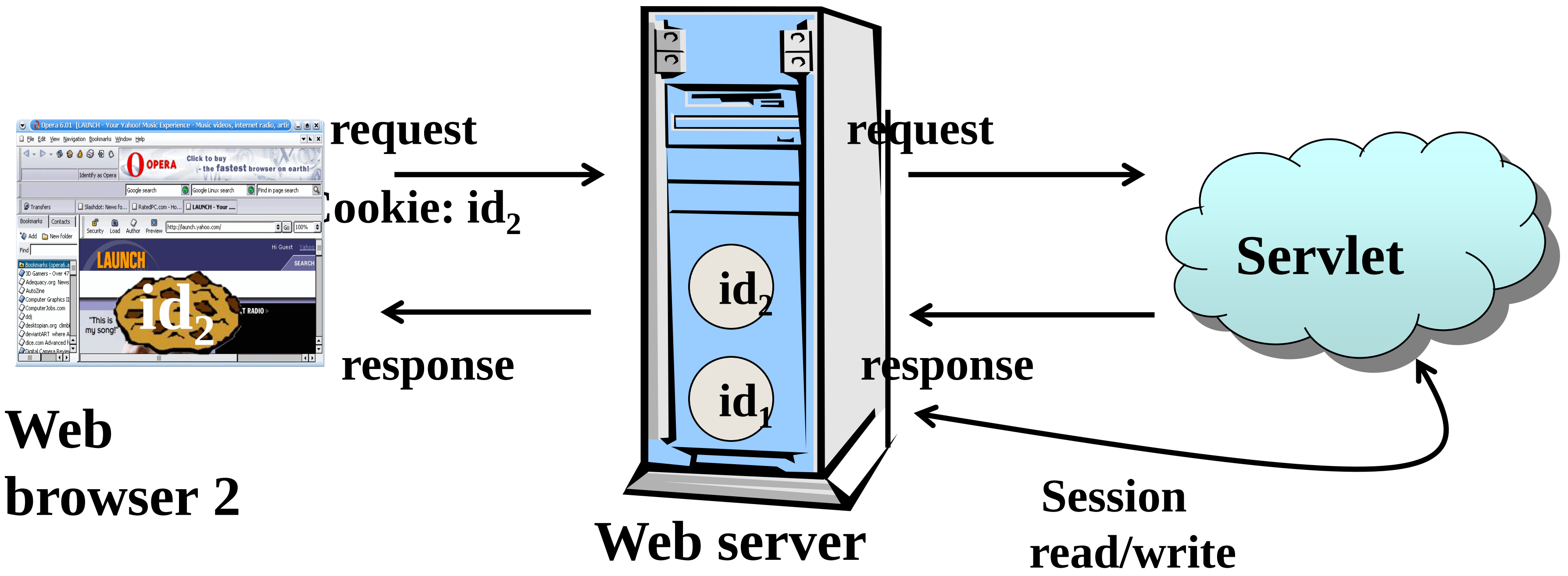
Session Cookies



Session Cookies



Session Cookies



Create Cookies

- To create cookies you need to use Cookie class of javax.servlet.http package.
- **Syntax**
- Cookie c=**new** Cookie(name, value);
- // here name and value are string type

Add Cookies

- To add a cookie to the response object, we use
- addCookie() mehtod.
- **Syntax**
- Cookie c=**new** Cookie(); //creating cookie object
  response.addCookie(c1); //adding cookie in the response

Read Cookies for browser

- To read Cookies from browser to a servlet, we need to call getCookies methods given by request object and it returns an array type of cookie class.
- **Syntax**
- response.addCookie(c1);
- Cookie c[]=request.getCookie();

HTML CODE

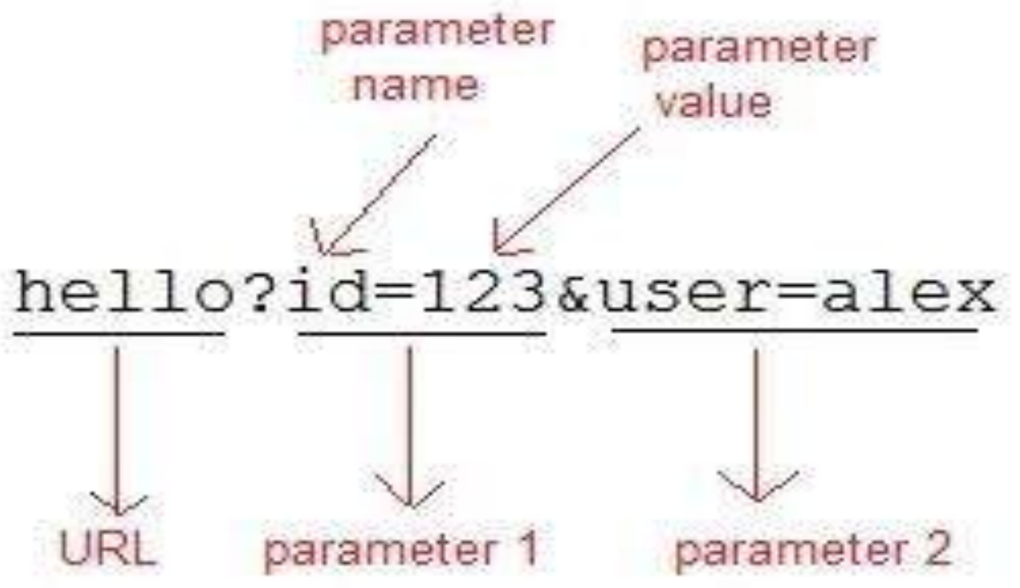
- <form action="FirstServlet" method="post">
- Name:<input type="text" name="userName"/> <br/>
- <input type="submit" value="continue"/>
- </form>

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
 public class FirstServlet extends HttpServlet {

 public void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException , IOException {
    response.setContentType("text/html");
   PrintWriter out = response.getWriter();

   String n=request.getParameter("userName");
   out.print("Welcome "+n);

   Cookie ck=new Cookie("uname",n);//creating cookie object
   response.addCookie(ck);//adding cookie in the response

   out.print("<form action='SecondServlet' method='post'>");
   out.print("<input type='submit' value='continue'>");
   out.print("</form>");
         out.close();
      } }
```

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
  public class SecondServlet extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse
     response)throws ServletException , IOException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    Cookie ck[]=request.getCookies();
    out.print("Hello "+ck[0].getValue());
    out.close();
    }    }
```

URL rewriting:

- URL rewriting is a way of appending data at the end of URL. Data is appended in name value pair form. Multiple parameters can be appended in one URL with name value pairs.

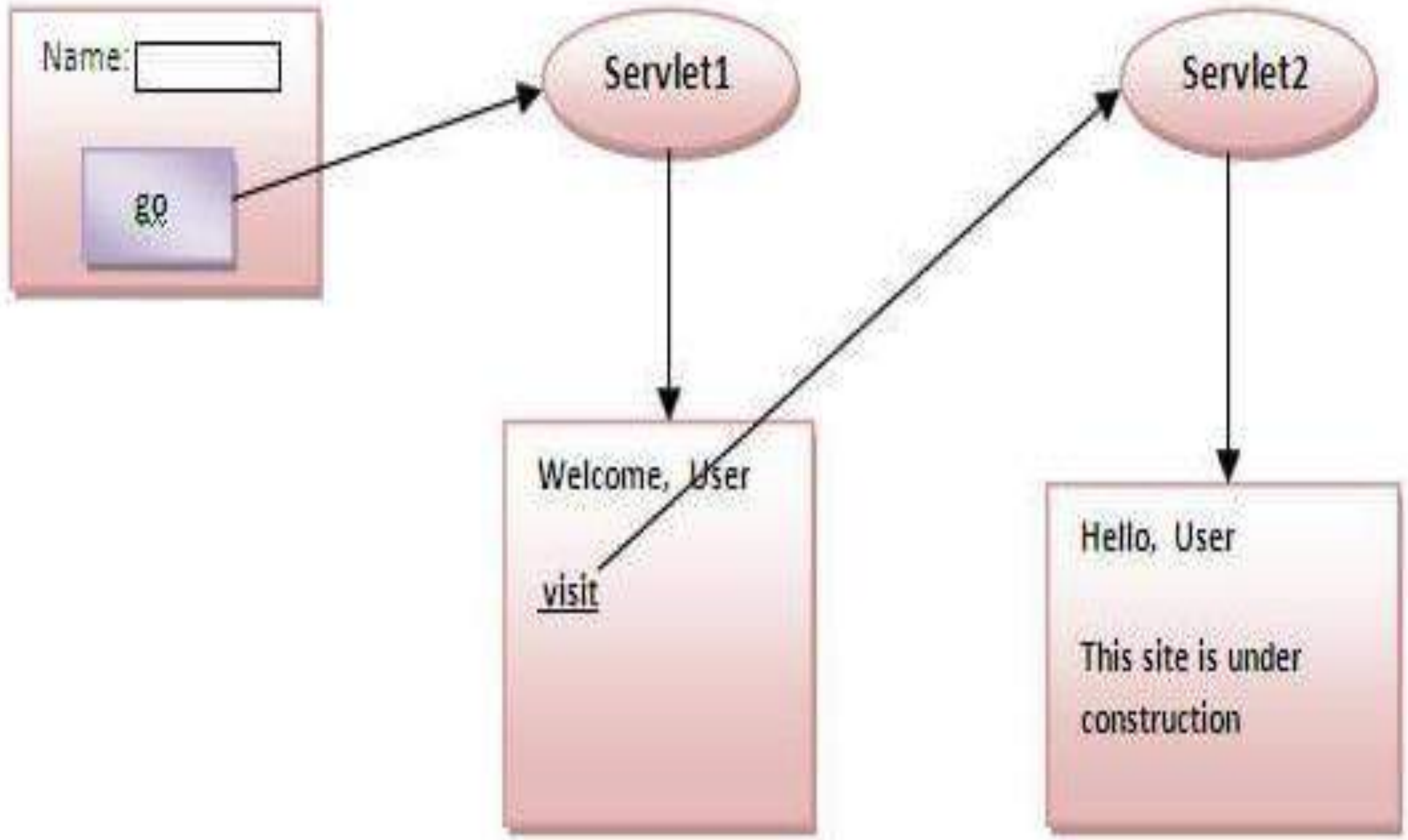


How to get parameter value from url in servlet?

- HttpServletRequest interface’s getParameter() method is used to get parameter value from url in servlet.
- ***Syntax:***
- String value = request.getParameter(“fieldName”);

Advantages of URL rewriting:

- As data is appended in the URL it is easy to debug.
- It is browser independent.
- ***Limitations of URL rewriting:***
- Not secure because data is appended in the URL.
- Can’t append large no. of parameters because URL length is limited.


Name:
go
Servlet1
Servlet2
Welcome, User
visit
Hello, User
This site is under construction

HTML CODE

```
<form action="Servlet1">
        Name <input type="text" name="uname">
        <input type="submit" value="submit">
</form>
```

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

  public class Servlet1 extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response) ) throws
      ServletException , IOException {
response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String n=request.getParameter("uname");
        out.print("Welcome "+n);

        //appending the username in the query string
        out.print("<a href='servlet2?uname="+n+"'>visit</a>");

        out.close();
}
}
```

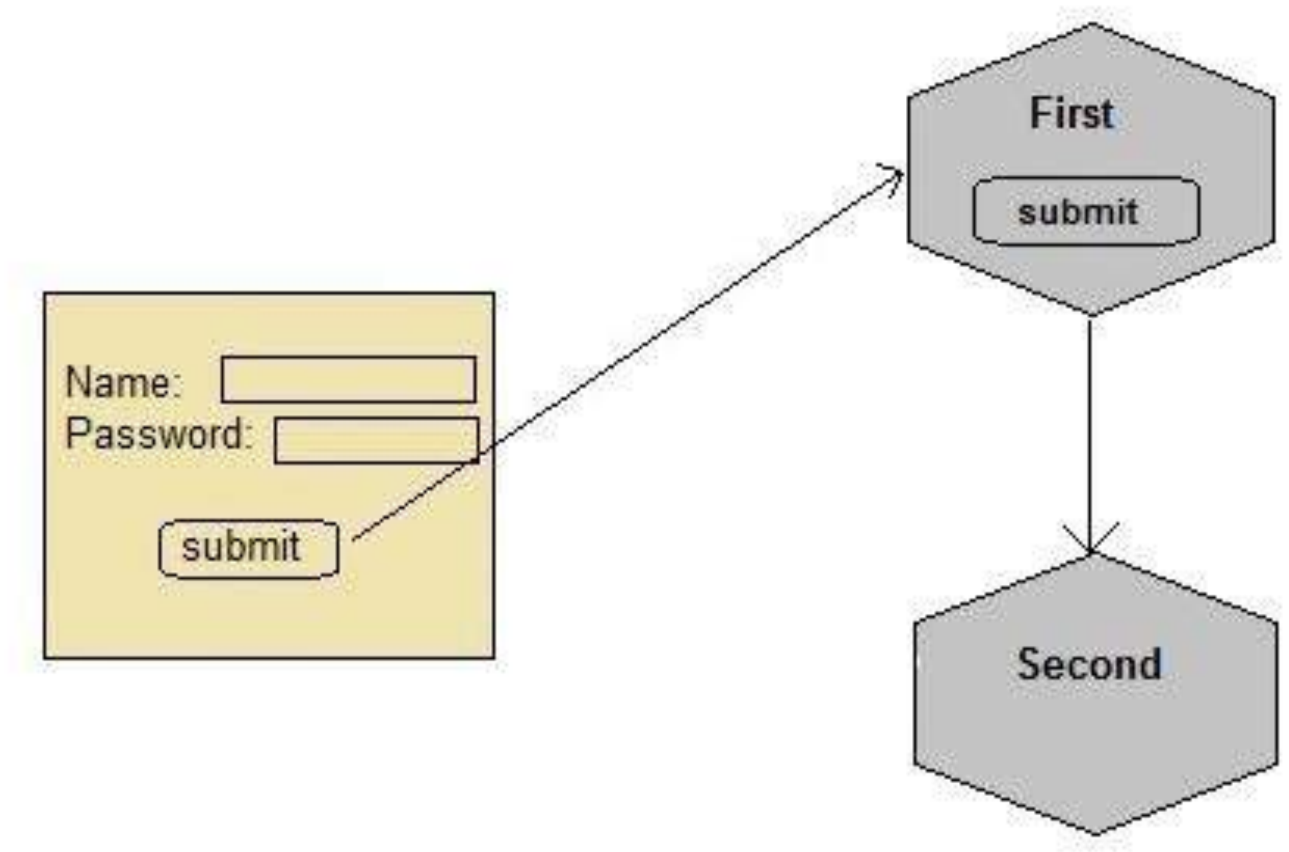
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Servlet2 extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response) ) throws
    ServletException , IOException {
response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      //getting value from the query string
      String n=request.getParameter("uname");
      out.print("Hello "+n);

      out.close();
 }
 }
```

Hidden Form Field

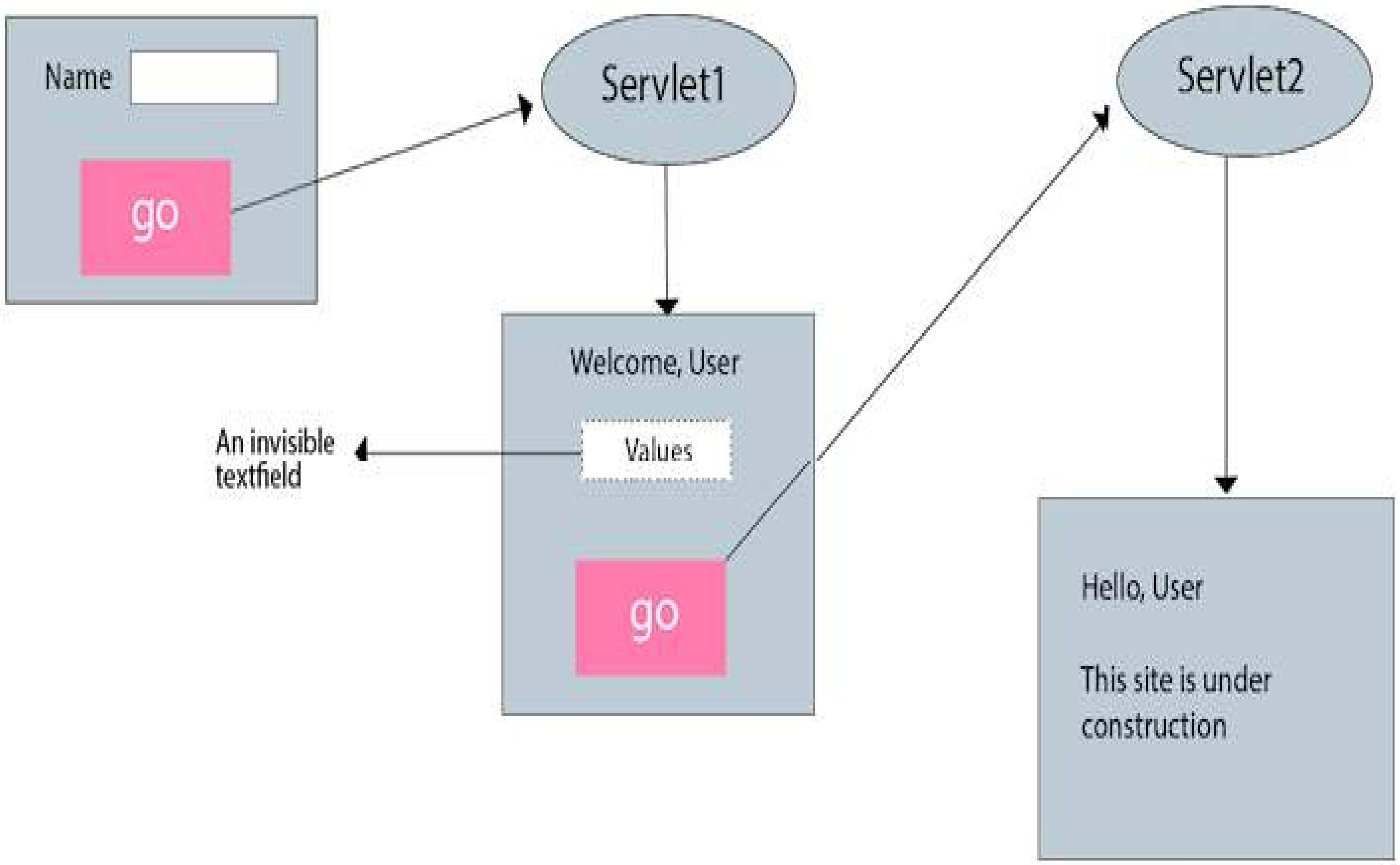
- Tracking client conversion using Html hidden variables in secure manner is known as hidden form field.
- **How to use Hidden Form Field ?**
- Syntax of Hidden field is:
- **<input type="hidden"> and with this we assign session ID value.**
- **Syntax**
- **<input** type="hidden" name=“sname" value=“mca"**>**

Hidden Form Field

- In case of Hidden Form Field **a hidden (invisible) textfield** is used for maintaining the state of an user.
- In such case, we store the information in the hidden field and get it from another servlet. This approach is better if we have to submit form in all the pages and we don't want to depend on the browser.
- Real application of hidden form field
- It is widely used in comment form of a website. In such case, we store page id or page name in the hidden field so that each page can be uniquely identified.


Name:
Password:
submit
First
submit
Second

In this example, we are storing the name of the user in a hidden textfield and getting that value from another servlet.



Hidden Form Field Advantage

- Basic knowledge of html is enough to work with this technique.
- It will always work whether cookie is disabled or not.
- Hidden boxes resides in web pages of browser window so they do not provide burden to the server.
- This technique can be used along with all kind of web server or application server.

Hidden Form Field Limitations

- More complex than URL Rewriting.
- It is maintained at server side.
- Extra form submission is required on each pages.
- Hidden form field can not store java object as values. They only store text value
- It Also increase network traffic because hidden boxes data travels over the network along with request and response.
- Hidden boxes does not provides data security because their data can be view through view source option.

HTML FILE

- **<form** action="FirstServlet ">
- Name:**<input** type="text" name="userName"/>
- **<br/>**
- **<input** type="submit" value="continue"/>
- **</form>**

```
mport java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
  public class FirstServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response){

          response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String n=request.getParameter("userName");
        out.print("Welcome "+n);
                    out.print("<form action='servlet2' >");
        out.print("<input type='hidden' name='uname' value='"+n+"'>");
        out.print("<input type='submit' value='continue'>");
        out.print("</form>");
        out.close();
  }
           }
```

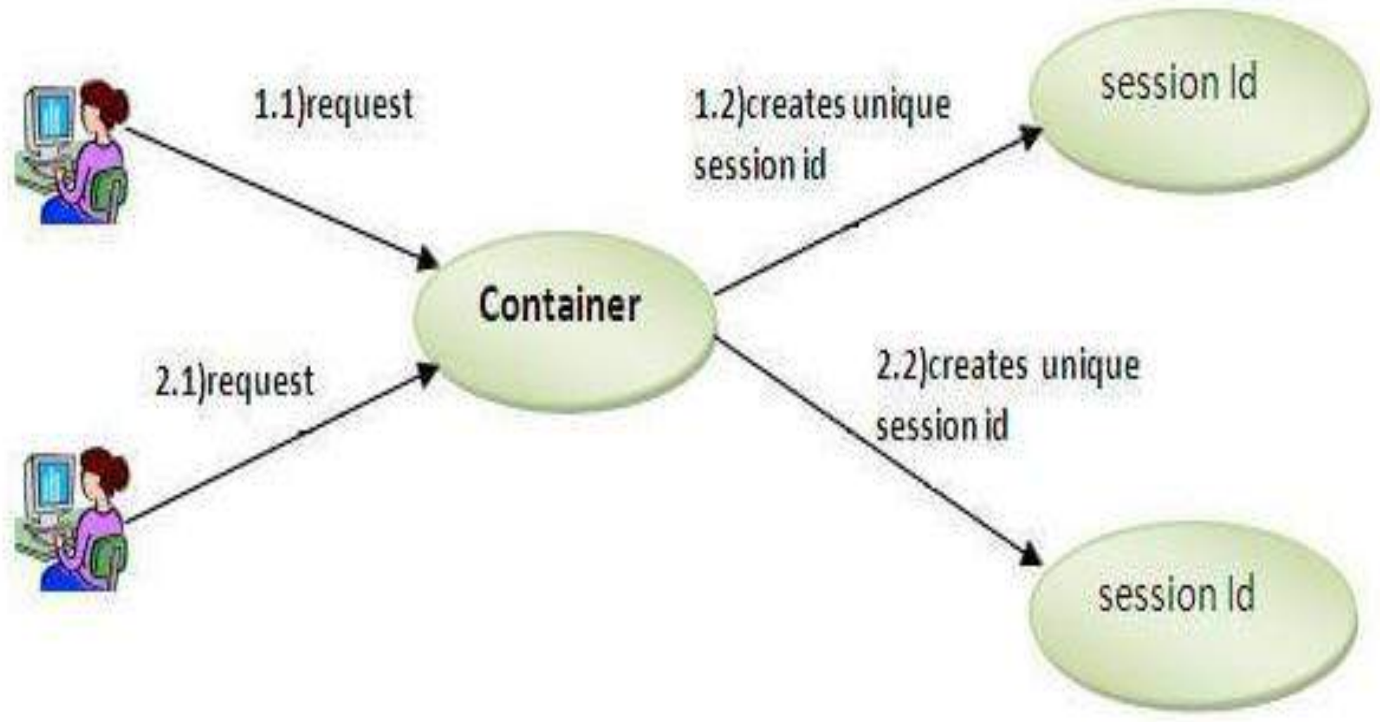
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SecondServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
    try{
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    //Getting the value from the hidden field
    String n=request.getParameter("uname");
    out.print("Hello "+n);

    out.close();
        }catch(Exception e){System.out.println(e);}
  }
}
```

HttpSession interface

- In such case, container creates a session id for each user. The container uses this id to identify the particular user. An object of HttpSession can be used to perform two tasks:

I. bind objects

II. view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.


1.1)request
2.1)request
Container
1.2)creates unique session id
2.2)creates unique session id
session Id
session Id

How to get the HttpSession object

- The HttpServletRequest interface provides two methods to get the object of HttpSession:
- **public HttpSession getSession():**Returns the current session associated with this request, or if the request does not have a session, creates one.
- **public HttpSession getSession(boolean create):**Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

- Commonly used methods of HttpSession interface
- **public String getId():**Returns a string containing the unique identifier value.
- **public long getCreationTime():**Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
- **public long getLastAccessedTime():**Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.
- **public void invalidate():**Invalidates this session then unbinds any objects bound to it.

- < form action="login">
- User Name:<input type="text" name="userName"/> <br/>
- Password:<input type="password"name="userPassword"/>
- <br/> <input type="submit" value="submit"/> </form>

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MyServlet1 extends HttpServlet {
   public void doGet(HttpServletRequest request,
    HttpServletResponse response){
      try{
       response.setContentType("text/html");
       PrintWriter pwriter = response.getWriter();
   String name = request.getParameter("userName");
       String password = request.getParameter("userPassword");
       pwriter.print("Hello "+name);
       pwriter.print("Your Password is: "+password);
       HttpSession session=request.getSession();
       session.setAttribute("uname",name);
       session.setAttribute("upass",password);
       pwriter.print("<a href='welcome'>view details</a>");
       pwriter.close();
     }catch(Exception exp){
        System.out.println(exp);
      }
    }
   }
```

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MyServlet2 extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response){
  try{
      response.setContentType("text/html");
      PrintWriter pwriter = response.getWriter();
      HttpSession session=request.getSession(false);
      String myName=(String)session.getAttribute("uname");
      String myPass=(String)session.getAttribute("upass");
      pwriter.print("Name: "+myName+" Pass: "+myPass);
      pwriter.close();
  }catch(Exception exp){       System.out.println(exp);
   }
  }
```